НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки

Обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій Стіренко

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система моніторингу для розумного дому (серверна частина)»

Виконав:
студент IV курсу, групи ІО-63
Орихівський Євгеній Романович _____

Керівник:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
Доцент кафедри СКС,
Орлова Марія Миколаївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет Інформатики та обчислювальної техніки

Обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій Стіренко

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Орихівський Євгеній Романович

1. Тема проєкту «Система моніторингу для розумного дому (серверна частина)», керівник проєкту Сімоненко Валерій Павлович, професор, д.т.н, затверджені наказом по університету від 07 травня 2020р. №1081-с

2. Термін подання студентом проєкту 20 травня 2020р.

3. Вихідні дані до проєкту технічне завдання, теоретичні данні

4. Зміст пояснювальної записки проведення дослідження предметної області, дослідження веб-додатків, вибір технологій, інструкція користувача

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

Структурна схема системи, принципова схема, блок-схема алгоритму.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Норма контроль	Сімоненко В.П., професор, д.т.н.		

7. Дата видачі завдання 20.02.2020 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	01.09.2019	
2.	Аналіз завдання	25.03.2020	
3.	Пошук інформації і готових рішень	11.04.2020	
4.	Вибір технологій для реалізації системи	20.04.2020	
5.	Програмування системи	28.04.2020	
6.	Оформлення пояснювальної записки	07.05.2020	
7.	Захист програмного продукту	14.05.2020	
8.	Передзахист	26.05.2020	
9.	Захист	20.06.2020	

Студент Євгеній Орихівський

Керівник Валерій Сімоненко

Анотація

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, 3 додатки та 15 джерел – загалом 58 сторінок.

Об'єкт дослідження: Система моніторингу для розумного дому (серверна частина).

Мета дипломного проекту: збір інформації про концепцію односторінкових веб-додатків, переваги та недоліки в порівнянні з багатосторінковими, огляд і вибір фреймворка, пошук і перегляд відомих SPA додатків, порівняння технологій використаних для їх створення. Розібратися з налаштуванням серверу і спробувати створити власний веб-додаток з використанням фреймворку для збору даних з датчиків температури і вологи.

У першому розділі було досліджено концепції веб-додатків, клієнт-серверну архітектуру. Проаналізовано необхідність веб-додатків в сучасному світі.

У другому розділі розглянуто методи створення SPA та інструменти для розробки. Порівняння архітектури і роботи SPA з різними технологіями.

У третьому розділі описано технології використанні при розробленні веб-додатку. Проведено тестування системи в реальних умовах.

У четвертому розділі було наведено інструкцію користувача, і можливості використання системи.

Ключові слова: SPA, веб-додаток, клієнт-серверна архітектура

Annotation

The explanatory note of the diploma project consists of four sections, 3 appendices and 15 sources - a total of 58 pages.

The object of study: Smart home monitoring system (server side).

The aim of the diploma project: collecting information about the concept of one-page web applications, advantages and disadvantages compared to multi-page, review and selection of the framework, search and review of well-known SPA applications, comparison of technologies used to create them. Learn how to set up a server and try to create my own web application using a framework to collect data from temperature and humidity sensors.

The first section explores the concepts of web applications, client-server architecture. The necessity of web applications in the modern world is analyzed.

The second section discusses the methods of creating a SPA and tools for development. Comparison of architecture and operation of SPA with different technologies.

The third section describes the technologies used in developing a web application. The system was tested in real conditions.

The fourth section provides user instructions and options for using the system.

Keywords: SPA, web application, client-server architecture

Опис альбому
дипломного проєкту

на тему: «Система моніторингу для розумного дому (серверна частина)»

Київ – 2020

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6322 00.000 ВП	Відомість дипломного проєкту	1	
3	А4	ДП 6322 01.000 ТЗ	Технічне завдання	3	
4	А4	ДП 6322 02.000 ПЗ	Пояснювальна записка	58	
5	А3	ДП 6322 03.000 Д1	UML діаграма взаємодії користувача з системою Схема структурна	1	
6	А3	ДП 6322 04.000 Д2	Схема взаємодії компонентів системи Схема функціональна	1	
7	А3	ДП 6322 05.000 Д3	Алгоритм роботи системи Схема принципова	1	

				ДП 6322 00.000 ВП	
	ПІБ	Підп.	Дата		
Розробн.	Орихівський Є.Р.			Відомість дипломного проєкту	Лист / Листів
Керівн.	Сімоненко В.П.				
					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-63
Н/Контр.	Сімоненко В.П.				
Зав.каф.	Стіренко С.Г.				

Технічне завдання
до дипломного проєкту

на тему: «Система моніторингу для розумного дому (серверна частина)»

Київ – 2020

ЗМІСТ

				ДП 6322 01.000 ТЗ		
	ПІБ	Підп.	Дата			
Розробн.	Орихівський Є.Р.			Технічне завдання	Лист	Листів
Керівн.	Сімоненко В.П.				1	3
					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-63	
Н/Контр.	Сімоненко В.П.					
Зав.каф.	Стіренко С.Г.					

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку системи моніторингу для розумного дому (серверна частина). Область застосування: Веб-додаток для контролю температури і вологи.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут» Ім. Ігоря Сікорського.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка системи моніторингу для розумного дому (серверна частина) для контролю температури і вологи за допомогою веб-додатку.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література з розробки веб-додатків, публікації в Інтернеті з даної предметної області.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробляємого продукту

- Можливість доступу через веб-браузер
- Реалізовано відображення і експорт даних
- Зрозумілий інтерфейс
- Доступ до бази даних

5.2. Вимоги до програмного забезпечення

- Будь-яка операційна система з встановленим браузером
- Node.js 8 і вище
- Npm 5 і вище

5.3. Вимоги до апаратної частини

- Комп’ютер на базі процесору Intel Core i3 і вище
- Оперативної пам’яті не менше 4 Гбайт
- Доступ до інтернету

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	11.04.2020
Складання і узгодження технічного завдання	20.04.2020
Аналіз структури програмного забезпечення	25.04.2020
Створення модулів системи, що розробляється	01.05.2020
Тестування окремих модулів системи	05.05.2020
Доопрацювання, налагодження і виправлення помилок	08.05.2020
Оформлення документації дипломної роботи	15.05.2020

Пояснювальна записка
до дипломного проєкту
на тему: «Система моніторингу для розумного дому (серверна частина)»

Київ – 2020 року

Зміст

СПИСОК УМОВНИХ СКОРОЧЕНЬ

SPA - додаток на одній сторінці

MPA - багатосторінковий додаток (традиційний додаток, що завантажує нові сторінки при натисканні на посилання)

PWA - прогресивний веб-додаток (веб-сайт, який створений за допомогою JavaScript або його фреймворків і може діяти як додаток, тобто ви можете, наприклад, додати його на домашню сторінку свого мобільного телефону як додаток)

SaaS (software as a service) – це модель надання ліцензії на програмне забезпечення за передплатою

ВСТУП

Сьогодні не можливо уявити своє життя без технологій, вони повсюди, починаючи від смарт-годинників і закінчуючи автомобілями зі штучним інтелектом. Особливими стає група розумної побутової техніки, адже вона допомагає зробити наше життя простішим, а побут комфортнішим.

Нікому не секрет, що на сьогодні існують такі побутові прилади, як розумні замки, смарт-холодильники, смарт-чайники, розумні пральні машини, смарт-телевізори, смарт-розетки, роботи пилососи і тому подібні пристрої. Всіх їх об'єднує те, що практично в назві кожного з них присутнє слово "смарт". Що ж означає слово "смарт", і як його трактувати в реальному житті?

Взагалі це слово має декілька трактувань, але що стосується технологій, ІТ-сфери, то тут слово "смарт" береться як прямий переклад з англійської мови й означає "розумний". Це означає, що кожний прилад, в назві якого використано приставку "смарт", по суті своїй є штучним інтелектом, який вміє виконувати якісь запрограмовані дії, а в деяких випадках приймати важливі рішення, які допоможуть полегшити наше життя або навіть зберегти його.

Сьогодні все більше і більше людей задумуються про ідеї смарт-будинків, в яких їм не прийдеться задумуватися про те чи безпечно в їхньому житлі, чи прибрано в них в кімнатах, тепло чи холодно в залежності від пори року, чи свіжі та в достатній кількості продукти знаходяться в них в холодильнику. І такі будинки вже існують, але чи є вони доступними для кожної людини? Звичайно ж ні, адже технології не є дешевим задоволенням, бо потребують інвестицій для створення і тестування нових і нових технологій, з яких не всі втілюються в життя.

Хоча, є пристрої які мають занадто велику ціну, в порівнянні з затраченими ресурсами на їх виробництво. Для мене стало цікавим скільки можна зекономити, якщо виготовити пристрій, який не буде поступатися готовим рішенням, але коштуватиме вдвічі, а то і втричі дешевше. Для експерименту було вибрано пристрій для вимірювання температури і вологи в приміщенні і запис даних у веб-додатку, за допомогою якого можна в онлайн

режимі збирати покази з будь-якої точки світу.

Метою роботи буде збір інформації про концепцію односторінкових веб-додатків, переваги та недоліки в порівнянні з багатосторінковими, огляд і вибір фреймворка, пошук і перегляд відомих SPA додатків, порівняння технологій використаних для їх створення. Розібратися з налаштуванням серверу і спробувати створити власний веб-додаток з використанням фреймворку для збору даних з датчиків температури і вологи.

РОЗДІЛ 1.

1. ДОСЛІДЖЕННЯ КОНЦЕПЦІЙ ВЕБ-ДОДАТКУ

1.1. Концепції веб-додатку

У сучасному світі використання веб-додатків зростає з кожним днем. Професіонали програмного забезпечення, такі як розробник програмного забезпечення та тестувальники програмного забезпечення, повинні ознайомитися з веб-додатками[1].

Це прикладна програма клієнт-сервер, що зберігається на віддаленому сервері, яка використовує веб-браузери та веб-технології для виконання певних функцій через Інтернет та через інтерфейс браузера.

Як було сказано вище, це прикладна програма клієнт-сервер, тому в середовищі клієнт-сервер кілька комп'ютерів можуть обмінюватися такою інформацією, як збереження інформації в базі даних. "Клієнт" може бути використаний для введення інформації, а "сервер" використовується як сховище для інформації.

Простими словами, ви можете визначити його як комп'ютерну програму, яка виконує певні завдання у свого клієнта, використовуючи веб-браузер. Додатки, які базуються на мережі, також відомі як веб-додатки.

Поширені приклади включають: веб-пошту, таку як Gmail, Yahoo та AOL, роздрібні продажі в Інтернеті, онлайн-форми, торгові візки, текстові процесори, електронні таблиці, редагування відео та фотографій, перетворення файлів, сканування файлів, Google Apps, такі як Google Docs, Google Таблиці, Google Слайди, інтернет-сховища тощо.

Існує шість різних типів веб-додатків:

- Статичний
- Динамічний
- Інтернет-магазин або електронна комерція
- Портал веб-додатків
- Анімовані

- Система управління вмістом

Для розуміння поняття веб-програми потрібен веб-сервер для обробки запитів користувачів, сервер додатків для виконання необхідних завдань і база даних для зберігання інформації.

Нижче наведено коротке пояснення цього процесу:

- Користувач надсилає запит на веб-сервер через Інтернет, який використовує або веб-браузер, або інтерфейс користувача програми.
- Далі веб-сервер надсилає запит користувача на відповідний сервер веб-додатків.
- Сервер виконує задану функцію, таку як запит до бази даних або обробка запитуваних даних.
- Сервер пересилає дані користувача на веб-сервер із запитаною інформацією.
- Зрештою, веб-сервер виводить на екран потрібну інформацію користувача.

Клієнт-серверна архітектура спрощує роботу веб-додатку. Клієнт використовується для введення інформації, а сервер використовується для зберігання та отримання інформації. Він взаємодіє з додатками, середнім програмним забезпеченням та базами даних для роботи з кількома програмами разом. Сервер приймає запит користувача, який надіслав через браузер. Після цього браузер виконує ці файли та відображає запитувану сторінку користувачеві. Тепер користувач може взаємодіяти з веб-сайтом.

Він може виконувати певні функціональні можливості через Інтернет, використовуючи веб-браузери та веб-технології. Програми обробляють сховище та отримують інформацію за допомогою скриптів на стороні сервера, таких як PHP та ASP. Представити інформацію користувачеві є можливість за допомогою сценаріїв на стороні клієнта, таких як JavaScript та HTML.

Найкращий приклад веб-програми — електронна пошта, де Gmail і

Outlook від Microsoft надає клієнтам електронну пошту на базі веб-сторінок.

За допомогою цього можна зробити наступні дії:

- Відображати звіти у графічному форматі.
- Отримати інформацію про товар в Інтернеті.
- Робітники можуть отримувати інформацію, пов’язану із завданнями, через мобільні пристрої.
- Програма дозволяє віддаленим працівникам виконати завдання протягом визначеного часу.
- Клієнти можуть легко відстежувати замовлення та бюджет товарів.

Перерахуємо пункти, які описують роботу з процесом веб-додатків:

- Спочатку клієнт подає запит на сервер HTTP через HTTP.
- Далі веб-сервер надішле повідомлення до статичного сховища даних за допомогою статичного запиту даних.
- Веб-сервер відповідає на статичне сховище даних і переміщується на сервер додатків, використовуючи запит сервлетів, що включає веб-контейнер та інші сервіси.
- Потім запит сервлета витягує інформацію з сховища даних програми та відповідає веб-серверу.
- Зрештою, веб-сервер дає відповідь користувачеві за допомогою HTTP відповіді.

Перелічимо переваги веб-додатку:

- Він працює на різних типах платформ.
- Дані захищено та легко взяти резервну копію.
- Ви можете легко оновити додаток.
- Ви можете легко використовувати низькі характеристики ПК або смартфонів.
- Це усуває проблеми сумісності, оскільки користувач може отримати доступ до тієї ж версії.

- За допомогою SaaS, зменшується програмне піратство в веб-додатках на основі підписки.
- За допомогою веб-програми працівники можуть працювати з будь-якого місця, користуючись доступом до Інтернету.

Наведемо основні навички, необхідні для розробки веб-додатків:

- Обробляйте сховище та отримуйте інформацію за допомогою скриптів на стороні сервера, таких як PHP, ASP.Net та Ruby.
- Мови скриптування на стороні клієнта (JavaScript, HTML та CSS).
- Photoshop
- WordPress та SEO
- Навички чуйного дизайну
- Інструменти розробки, такі як IDE, редактори (Visual Studio, Eclipse)
- Веб-сервери (Apache, IIS)

Чому ми використовуємо або потребуємо веб-додатків у галузі програмного забезпечення[2]:

- Можна легко створити чуйний дизайн веб-додатків для кращого користування.
- Зберігання веб-програми може бути збільшено, оскільки існує хмарне сховище.
- Це зменшує комерційні витрати, оскільки там менше обслуговування та низькі вимоги до системи кінцевого користувача.
- Не потрібно ніякої установки, оскільки всі системи матимуть браузер; таким чином усувається обмеження простору.
- Він розширює товари та послуги замовника, збираючи наявні відгуки клієнтів.
- Використовуючи веб-програми, ви контактуєте з будь-ким і будь-де в світі.

Обсяг цього включає підвищення продуктивності та ефективності замовника. Він використовує менші витрати бізнесу та дозволяє отримати доступ до ділової інформації в будь-якій точці світу. Це економить час та гроші, а також зв'язок між споживачами та діловими партнерами.

Ці технології мають велику цільову аудиторію, оскільки вони використовують як веб-браузери, так і веб-технології для виконання деяких функціональних можливостей через Інтернет. Її в основному використовують ділові особи, роздрібні продавці, розробники вікі, розробники сервісів обміну повідомленнями та багато іншого.

Ця технологія забезпечує хороший кар'єрний ріст. Щоб отримати магістра в цій галузі, вона потребує щоденної більшої практики та потребує більш гнучкості щодо всіх веб-тенденцій. Ця галузь найшвидше розвивається у сучасному світі, тому потрібно вивчати нові технології та бути більш динамічними.

1.2. Односторінкові застосунки (SPA)

Односторінковий додаток[3] - це додаток, якому не потрібно перезавантажувати сторінку під час її використання та працює в браузері. Подумайте про додатки, якими ви користуєтесь щодня: Facebook, Карти Google, Gmail, Twitter, Google Drive або навіть GitHub. Все це приклади SPA.

Однією з найкращих переваг правильно налаштованого SPA є користувацька робота (UX), де користувач насолоджується природним середовищем програми без необхідності чекати перезавантаження сторінки та інших речей. Ви залишаєтесь на тій же сторінці, яка працює на мові програмування JavaScript.

Основною перевагою односторінкових програм є його швидкість. Більшість ресурсів SPA (HTML + CSS + Scripts) завантажуються при запуску програми і не потрібно їх перезавантажувати під час використання. Єдине, що змінюється - це дані, які передаються на сервер і з нього. Як результат,

програма дуже добре реагує на запити користувача і не потрібно постійно чекати спілкування клієнт-сервер.

Численні дослідження Google та ключові висновки таких компаній, як Amazon, показують, що якщо на завантаження сторінки потрібно більше 200 мілісекунд, це може потенційно зруйнувати ваш бізнес або, щонайменше, коштувати вам великих грошей. Наприклад, Amazon каже, що 1 секунда додаткової затримки завантаження сторінки коштує їм 1% від продажів (що, враховуючи кількість продажів Amazon, становить 1,6 мільярда доларів на рік.)

З точки зору розробника, створення такого додатку впорядковується та оптимізується. Вам не потрібно писати код для візуалізації сторінок на сервері. Вам навіть не потрібен сервер, щоб розпочати процес розробки. Ви можете розпочати роботу з файлу, щоб розпочати роботу. Крім того, розробник може повністю використати той самий код на стороні сервера та ефективний API для веб-програми та нативного мобільного додатка.

Односторінкові програми є відмінними, коли у вас працює команда розробників, які працюють разом. Це дозволяє розробникам бекенда зосередитись на API, тоді як розробники інтерфейсів можуть приділяти більше уваги створенню найкращого користувацького досвіду на основі API бекенда та реалізації прекрасного інтерфейсу користувача.

Налагодження односторінкового додатка також просто за допомогою браузера Chrome, оскільки в ньому є спеціальні інструменти для Angular Batarang і React (технології, що застосовуються для SPA). За допомогою консолі можна відстежувати мережеві операції, а також досліджувати різні елементи сторінки та пов'язані з ними дані.

Процес кешування також досить ефективний - програма надсилає лише один запит, зберігає всі передані дані і може використовувати ці дані. Це особливо важливо в моменти, коли користувач може мати поганий зв’язок, але все ще може використовувати додаток, оскільки він синхронізований із сервером.

SPA можна легко перетворити на PWA. У свою чергу, це дає можливість розробникам надавати локальне кешування та пропонувати клієнтам та користувачам офлайн-досвід.

Незважаючи на всі переваги односторінкових програм, інколи SPA може бути не ідеальним рішенням.

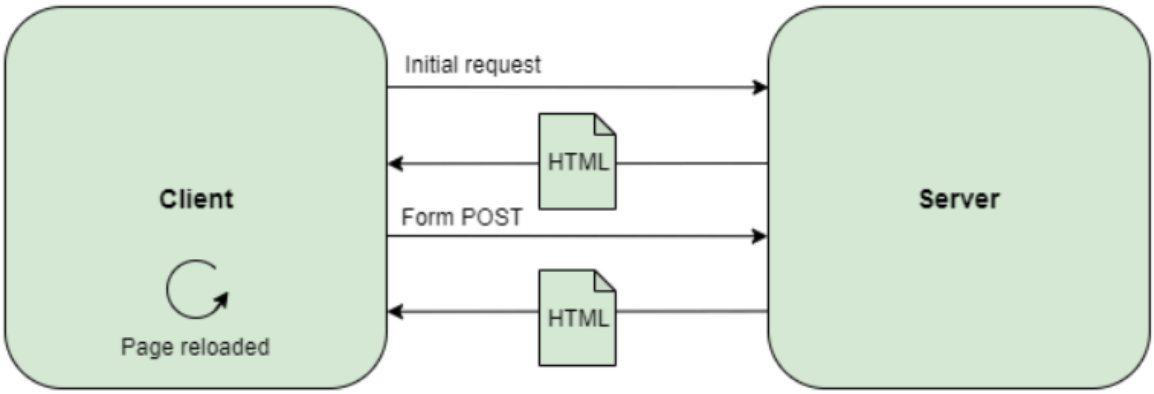


Рис. 1.1. MPA

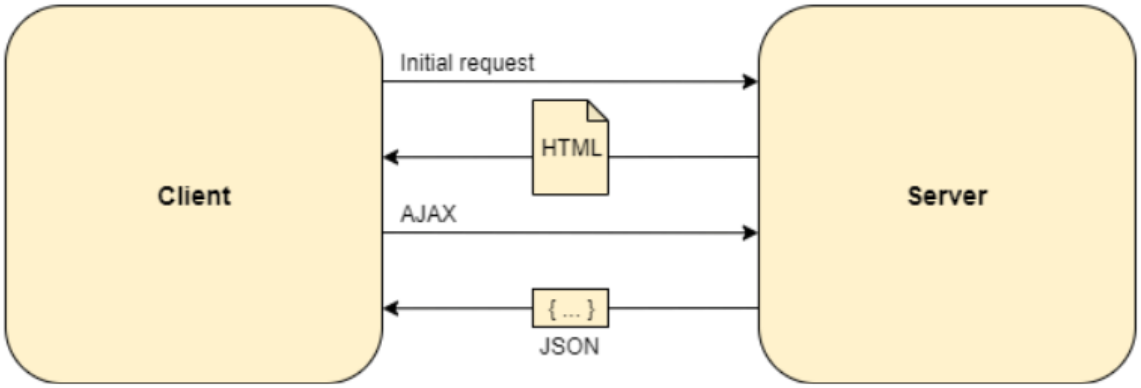


Рис. 1.2. SPA

Що мається на увазі під недоліком програми на одній сторінці?:

- Односторінкові програми додають на браузер навантаження. Наприклад, якщо ваші користувачі мають пристрої малої потужності, вони матимуть поганий досвід роботи програми з точки зору швидкості.
- Додаткові проблеми з JavaScript виникають, оскільки ви повинні переконатися, що немає витоків пам'яті. Оскільки додаток може працювати протягом тривалого часу (на відміну від MPA, де тривалість сторінки підраховується за лічені хвилини), вам

потрібно переконатися, що SPA не споживає більше пам'яті, ніж потрібно. Інакше задоволення від швидкого завантаження сторінок буде знищено млявістю недоступної пам’яті на пристрої користувача.

- Ще одним недоліком JavaScript є те, що користувачі можуть просто вимкнути його на своїх пристроях, і тоді вам потрібно продумати додаткові способи доступу до інформації на вашому веб-сайті чи вашому додатку без JavaScript.
- Ще одна важлива річ, яку слід пам’ятати про використання або побудову SPA - це безпека. Завдяки міжсайтовому сценарію (XSS) та тому, що не завантажуються нові сторінки, хакери можуть отримати доступ до вашого веб-сайту та вводити нові сценарії на стороні клієнта.
- Тому, якщо ви думаєте про створення односторінкової програми для вашого бізнесу, переконайтеся, що ви вжили необхідних заходів, щоб цього не допустити.
- Ще одне питання безпеки - конфіденційність конфіденційних даних. Початкове завантаження сторінки не повинно містити будь-якої інформації, яка не повинна бути доступною для всіх користувачів. Оскільки весь SPA завантажується відразу на пристрій користувача, ви можете випадково подати дані, які повинні бути за входом або взагалі недоступними.

Безпечніше використовувати SPA, коли ви не розраховуєте на оптимізацію пошукових систем (SEO), наприклад, якщо ваш додаток доступний лише за допомогою входу.

Якщо є блог, з іншого боку, пошукові системи важче індексувати ваш SPA-сайт на відміну від традиційних сторінок, що надаються сервером. Це відбувається тому, що URL-адреса насправді не змінюється, а різні сторінки зазвичай не мають своїх URL-адрес.

Тому, якщо хочеться турбуватися з налаштуваннями URL-адрес, виберіть рамку, яка підтримує рендеринг на стороні сервера (SSR). Крім того, не використовуйте вікна iframe, статичні URL-адреси та оптимізуйте сценарії на своїй сторінці, щоб пришвидшити їх. Нарешті, переконайтеся, що у ваших сторінках є HTML5 для того, щоб сканер Google мав доступ до них.

Також слід звернути особливу увагу на серверні повідомлення, особливо на помилки 200 та 404.

Постає питання: "Коли я повинен використовувати програму на одній сторінці?" Коли у вас є бізнес або особистий веб-сайт, який потребує динамічної платформи та невеликого обсягу даних, односторінковий додаток - це гарна ідея. Це також чудовий варіант, якщо ви плануєте розробляти мобільний додаток у майбутньому, оскільки, як ми вже згадували вище, бекенд API можна використовувати як для Інтернету, так і для мобільних додатків.

Основним недоліком є SEO, але архітектура підходить для платформ Software-as-a-Service (SaaS), закритих спільнот та соціальних мереж (саме тому Facebook використовує це). Причиною цього є те, що цим веб-сайтам не потрібна оптимізація для пошуку в Google.

Великі компанії з широким спектром послуг та продуктів отримали б перевагу від традиційного багатосторінкового додатку. До таких підприємств належать інтернет-магазини, сайти компаній, каталоги та торгові майданчики. Керування таким додатком також було б набагато простіше, оскільки можливо підключити його до декількох баз даних користувачів.

Нарешті, для таких типів компаній знадобляться варіанти оптимізації пошукових систем, оскільки багато з них хочуть знайти їх у Google, Bing та інших пошукових системах (щоб продати пропоновані ними продукти чи послуги.)

Основна перевага односторінкових програм для кінцевих користувачів, звичайно, швидкість, з якою завантажується програма, а також можливість працювати в автономному режимі.

SPA пропонує лінійний досвід користувача, а це означає, що легко переходити на веб-сайт і розуміти, де знайти ту чи іншу річ. Наприклад, SPA Saucony має чіткий початок, середину та кінець. Використовуючи дизайн UI / UX, розробники додатків Saucony використовували прокрутку паралакса та переходи, щоб зробити подорож клієнтом приємною.

Односторінкові програми також чудово підходять для мобільних пристроїв, оскільки більшість часу всі користувачі потребують прокрутки (подумайте про нескінченну стіну Facebook). Вам не потрібно натискати жодних посилань, і ви просто насолоджуєтесь прокручуванням.

Що стосується бізнесу, розробка додатків на одних сторінках зазвичай займає менше часу, оскільки один і той же запуск API може використовуватися як для Інтернету, так і для мобільних пристроїв. В результаті потік інформації впорядковується, і створювати автономний мобільний додаток стає набагато простіше.

Беручи до уваги переваги оптимізації швидкості SPA, бізнес також може отримати користь, оскільки користувачі будуть зацікавлені у використанні швидкого додатка.

ВИСНОВКИ ДО РОЗДІЛУ 1

В ході виконання даного розділу дипломного проєкту, було досліджено концепції веб-додатків, клієнт-серверну архітектуру. Проаналізовано необхідність веб-додатків в сучасному світі. Розглянуто переваги і недоліки SPA у порівнянні з MPA. Стало зрозумілим чому ми використовуємо або потребуємо веб-додатків у галузі програмного забезпечення. Більш детально розібрався з тим, коли і навіщо потрібно використовувати SPA.

Зробив висновок, що свій веб-застосунок найкраще реалізувати за допомогою SPA. Це дасть мені приємнішу взаємодію з сайтом і швидкодію моїй системі в цілому.

РОЗДІЛ 2.

2. МЕТОДИ СТВОРЕННЯ SPA ТА ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ

2.1. Архітектура і робота SPA

Як працюють односторінкові програми? Односторінкова архітектура додатків досить проста - вона складається з клієнтських технологій (зазвичай React.js, Angular та Vue.js) та серверних технологій (для яких в більшості випадків використовується Node.js)

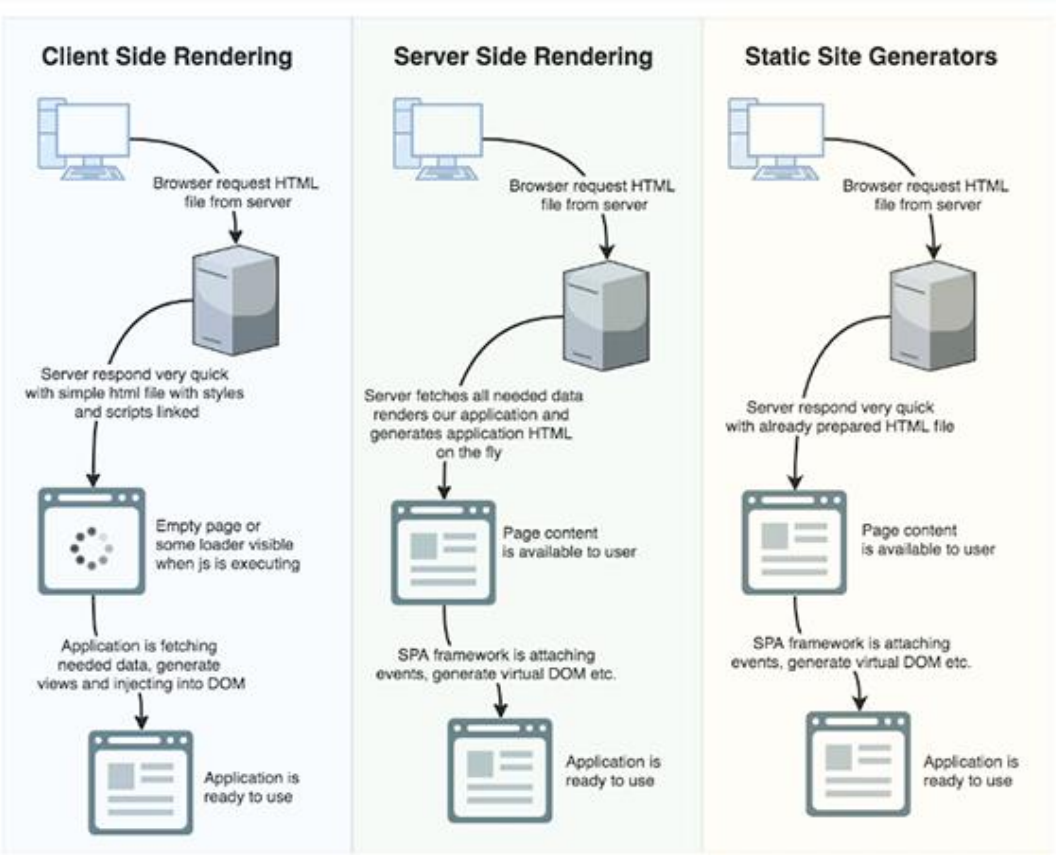


Рис. 2.1. Точки зору способу відображення сайту під час його представлення користувачеві.

2.1.1. Client-side rendering (CSR)

CLIENT-SIDE RENDERING (CSR):

- Браузер надсилає запит на HTML-файл із сервера
- Сервер швидко реагує на простий HTML-файл із пов'язаними стилями та сценаріями
- Користувач бачить порожню сторінку або зображення

завантажувача під час виконання js

- Додаток отримує дані, генерує представлення даних та вводить їх у DOM
- Додаток готовий до використання

Це може бути варіантом для простих веб-сайтів, але потрібно мати на увазі, що для надання інформації на стороні клієнта потрібно багато ресурсів з пристрою і можливість перевантажити браузер. Як результат, цей варіант може бути найповільнішим із трьох. У той же час, якщо у вас веб-сайт з високою трафіком, КСВ буде кращим, оскільки він буде представляти інформацію користувачеві, не надто розмовляючи з сервером.

Крім того, якщо вам потрібні параметри соціального обміну, майте на увазі, що всі сторінки в CSR зазвичай мають однакові відкриті графіки, тому вам потрібно використовувати або SSR, або SSG.

2.1.2. Server-side rendering (SSR)

SERVER-SIDE RENDERING (SSR):

- Браузер надсилає запит на HTML-файл із сервера
- Сервер отримує всі необхідні дані, надає програму та генерує HTML-файл програми на ходу
- Користувач бачить доступний вміст
- Односторінкова рамка програми - це те, що приєднує події, створює віртуальний DOM та виконує інші дії
- Додаток готовий до використання

Відображення на стороні сервера - це варіант, на який є найкращим, оскільки він поєднує в собі швидкість односторінкової програми та не перевантажує браузер користувача, що робить додаток швидким.

2.1.3. Static site generators (SSG)

STATIC SITE GENERATORS (SSG):

- Браузер надсилає запит на HTML-файл із сервера
- Сервер швидко реагує на вже підготовлений HTML-файл

- Користувач бачить сторінку
- Додаток отримує дані, генерує представлення даних та вводить їх у DOM
- Додаток готовий до використання

Хороший і швидкий варіант, але потрібно пам’ятати, що якщо є динамічний контент на веб-сайті, статичні генератори сайтів не стануть найкращими друзями, оскільки вони орієнтовані більше на статичні сторінки.

2.2. SPA використовуючи Angular, React.js і Vue.js

Розробники випробували численні фреймворки та мають різний досвід роботи, але вони об'єднуються, сказавши, що для фронтенду є три найкращі рамки - Angular, React і Vue - і одна з найкращих для бекенда - Node.js. На ринку є кілька інших фреймворків (наприклад, Ember або Knockout), але їх використовують не так багато.

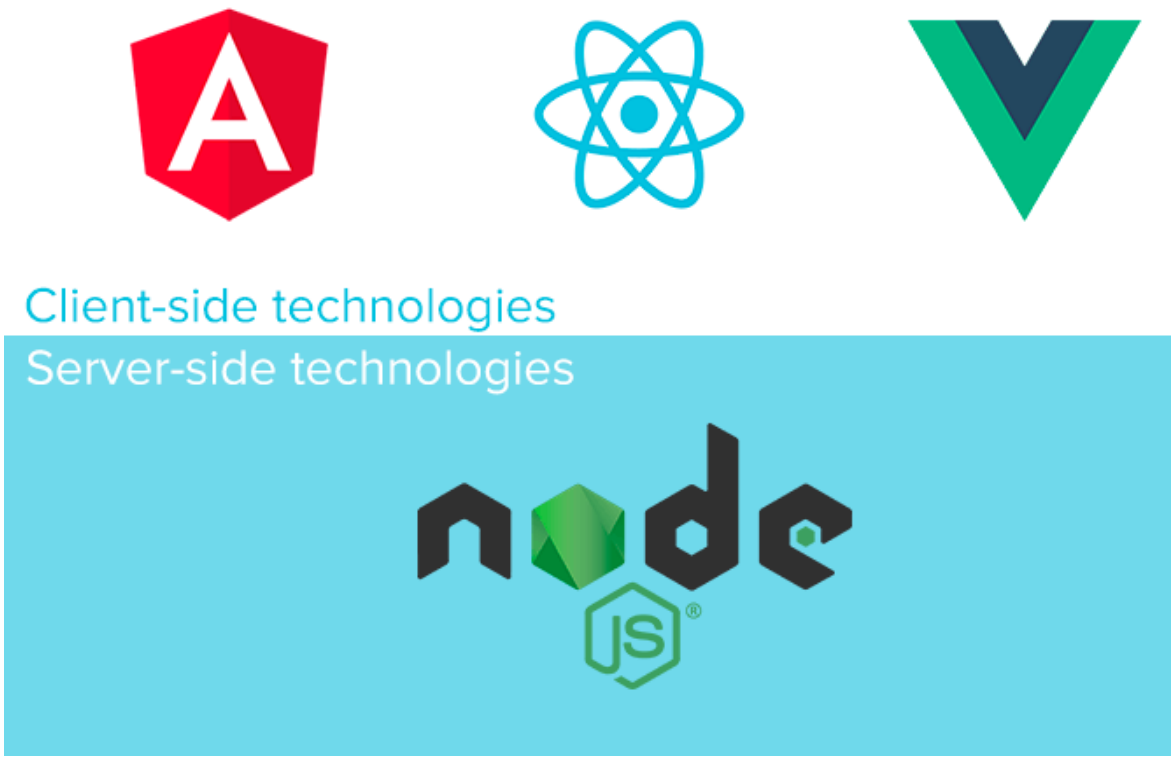


Рис. 2.2. Клієнт-серверні технології.

Щоб створити програму на одній сторінці, потрібні AJAX та HTML5 для

створення чуйних сторінок, в той час як Angular, React і Vue відповідальні за обробку «важкого підйому» на клієнтській базі SPA.

2.2.1. Angular і Single-Page Applications

Angular[4] - це структура JavaScript, яка була представлена компанією Google ще в 2010 році. Серед трьох варіантів, які ми порівнюємо, Angular є найстарішим, і він заснований на TypeScript. Завдяки TypeScript, Angular є чудовим варіантом використання великими командами розробників та компаніями, які вже використовують цю технологію в інших своїх продуктах.

Це найзріліший серед фреймворків і має пристойну кількість учасників GitHub. Це може бути складніше з точки зору кривої навчання, але в той же час воно того варте.

Переваги Angular:

- Незалежний від платформи. З AngularJS, ваш односторінковий додаток буде сумісний з кожною платформою. Це, безумовно, не зашкодить продуктивності, оскільки додатки, розроблені за допомогою AngularJS, є прогресивними, оптимізованими для високопродуктивної і нульової зупинки установки. Крім того, ваш односторінковий додаток (SPA) буде мати власні мобільні додатки зі стратегіями від Ionic Framework, NativeScript і React Native.
- Велика продуктивність. Розроблені в AngularJS, односторінкові програми будуть завантажуватися швидко. Секретний соус — це компонентний маршрутизатор, який забезпечує автоматичне розділення коду, коли користувачі завантажують тільки код, необхідний для перегляду їх запиту.
- Відмінна підтримка користувацького інтерфейсу. Коли справа доходить до користувача інтерфейсу та користувацького досвіду, односторінковий додаток повинний бути захоплюючим і надавати приємний досвід користування. AngularJS дає таку можливість.
- Хороша продуктивність. AngularJS живиться від надійної функції

тестування, яка допоможе перевірити додаток однієї сторінки з кожного кута. Ця структура допомагає дізнатися, якщо щось зламалось, і дає можливість швидко виправити поломку. Крім того, транспортир дозволяє виконувати тести сценаріїв швидше і стабільніше.

- Рясна Гнучкість. Після того, як у вас є AngularJS у вашому арсеналі, створення однієї сторінки додатка стає простіше, ніж коли-небудь. Платформа надає можливості для створення високопродуктивних, складних хореографій та анімаційних графіків. Все це без написання великої кількості коду, інтуїтивно зрозумілим Angular. Завдяки API-інтерфейсу можна буде мати величезну гнучкість для розробки інтерактивних, зручних для користувача і незалежних від платформи веб-додатків.
- Простота обслуговування. Розробка односторінкової програми з AngularJS допоможе обслуговувати його в майбутньому. Фреймворк настільки добре побудований, що коли досвідчений AngularJS розробник розробляє на ньому, то програми виявляються простими в обслуговуванні.

Серед клієнтів, які використовують Angular для своїх додатків на одній сторінці, є Google (та їхні продукти, такі як Gmail та Google Drive) та Wix, тому якщо ви вирішите створити програму для однієї сторінки за допомогою Angular, ви будете в чудовій компанії.

2.2.2. React.js and single-page applications

React.js[5] - бібліотека JavaScript, створена в 2013 році Facebook. Вони широко використовують його у всій своїй продуктовій лінійці, включаючи такі відомі додатки на одній сторінці, як сам Facebook, Instagram та WhatsApp. Крім Facebook, Uber також використовує бібліотеку React.js для своїх продуктів.

З трьох конкурентів, React має найбільшу кількість учасників GitHub - понад тисячу - що допомагає бути в курсі різних проблем, з якими стикаються розробники щодня.

Розробники також віддають перевагу React на проекти, які розробляють, якщо немає вимог до іншої основи. Він і легкий, і зрілий, щоб перевірити і випробувати його.

React - це правильний вибір для тих, хто тільки починає працювати з фронтальними рамками JavaScript, а також для стартапів та розробників, яким подобається бути гнучкими. React пропонує хороші варіанти інтеграції з різними іншими технологіями та рамками, що дуже корисно, коли ви працюєте над проектом з великою екосистемою.

React в основному використовувався для візуалізації уявлень у веб-додатках або мобільних додатках. Це дозволило розробникам створювати повторно використовувані компоненти, які не залежать один від одного. Тому, коли якась критична функція веб-додатки ламалася, їм все ще було краще з іншими елементами. Крім того, React приніс цю фантастичну функцію під назвою Virtual DOM, яка дозволила розробникам реалізувати SSR без необхідності оновлювати весь вигляд кожного разу під час оновлення.

Переваги React і чому ми повинні використовувати його для своїх проектів веб-додатків[6]:

- Легкий у вивченні для розробників. Одна з основних проблем розробників — це вибір фреймворку (або бібліотеки), який легше вивчити і реалізувати. React легко зрозумілий для розробників, які знайомі з Javascript. Тому, якщо у вас є команда розробників, які дуже добре розбираються в Javascript, Reactjs повинен бути вашим найкращим вибором. Однак, навіть якщо розробники не знають Javascript, React може бути правильним місцем для початку. На відміну від кутового, React має плавну криву навчання.
- React дозволяє розробникам використовувати компоненти. У React ваш додаток складається з компонентів. В ідеалі, ви починаєте з побудови невеликих компонентів, таких як кнопки, прапорці, випадаючі списки, меню і так далі. І все що потрібно, це створити компоненти оболонки навколо цих менших компонентів.

І оскільки ви продовжуєте писати компоненти оболонки більш високого рівня, у вас буде один кореневий компонент і кілька ієрархічних компонентів. Тепер, тут немає мізків: кожен компонент React має свою власну логіку. Тому, якщо ви хочете повторно використовувати компонент button через свій додаток, ви можете це зробити. Я майже впевнений, що всі хочуть повторного використання у своєму проекті.

- Він забезпечує унікальний рівень абстракції. Ще одне менш відома перевага, пов'язана з бізнесом, з допомогою React полягає в тому, що він дозволяє створити хороший рівень абстракції, що означає, що кінцевий користувач не може отримати доступ до складних внутрішніх пристроїв. Розробник повинен знати тільки деякі основи, і їм краще знати внутрішні функціональні можливості. Крім того, він не диктує ніяких архітектурних шаблонів, таких як MVC, MVP і MVVM. Розробник може вільно створювати архітектуру програми будь-яким способом, який він вважає за потрібне.
- Він добре зарекомендував себе з яскравою екосистемою інструментів розробника. React складається з багатої і яскравої екосистеми. Розробники можуть знайти десятки готових і параметризованих діаграм, графіків, інструментів, документації та інших компонентів, які дозволяють їм побудувати веб-додаток за менший час, не винаходячи велосипед. Існує дивовижна колекція Reactjs dev tools і підручники, які допомагають розробникам створювати приголомшливі матеріали.

Reactjs блищить у створенні динамічних і привабливих веб-інтерфейсів і торжествує над іншими JavaScript-фреймворками (такими як Angular, Ember). Причина полягає в наступному: Virtual DOM полегшує оновлення компонентів всякий раз, коли користувач виконує будь-яку взаємодію, не зачіпаючи інші частини інтерфейсу.

2.2.3. Vue.js and single-page applications

Vue - наймолодший, створений у 2014 році колишнім співробітником Google Yuxi (Evan) You. Незважаючи на те, що жодна велика корпорація не підтримує її розвиток, Vue зуміла підвищити популярність. В даний час такі компанії як GitLab, Baidu та Alibaba використовують Vue.js для своїх потреб.

Для побудови користувацьких інтерфейсів Vue — це найвідоміша і найкраща прогресивна платформа. Це платформа з відкритим вихідним кодом, яка в основному використовується для однієї сторінки. Особливо стартапи воліють використовувати Vue.js для доступного застосування сторінки, і великі компанії також використовують його і отримують користь.

Переваги Vue.js:

- Компактний і зрозумілий. Для зберігання розділених даних, користувальницьких методів і методів життєвого циклу за допомогою Vue.js існує чітка архітектура веб-проекту, користувач з легкістю додає Vue.js. Щоб зробити процес побудови сучасним веб-додатком, вітерець, спостерігачі, директиви і властивості з'єднання — це великі можливості Vue.js. Ви можете заощадити багато часу, і за допомогою цієї платформи ви також можете розробляти невеликі і великомасштабні веб-додатки. Для користувача не займе багато часу, щоб використовувати його і скачати, тому що достатньо малий розмір Vue.js 18-21 KB є значною перевагою його.
- Гнучкий і добре документований. За допомогою Vue.js можна розробляти крос-платформні додатки, використовуючи компоненти і пов'язуючи їх між собою. Використовуючи віртуальні вузли, чисті файли JavaScript і HTML, шаблони запису JavaScript і широкий спектр різних середовищ, ви можете отримати доступ до того, що надається Vue.js. Для функцій маршрутизації та управління станом, може бути використана функція Vuex, а також для більш великих і складних завдань.

- Можливості інтеграції. На основі JavaScript framework Vue.js є і на JavaScript, інші вбудовані програми розробників можуть бути інтегровані, саме тому з існуючими додатками для інтеграції Vue.js полегшує роботу розробників. Також, Vue.js має компоненти — це означає, що зміна існуючих і розробки нових додатків є корисними.
- Двосторонній зв'язок. Належний до своєї архітектурі MVVM, відрегулювані блоки HTML, роблять його легким у двосторонніх повідомленнях Vue.js assist. З цієї причини може здатися, що він дуже близький до Angular.js, що також прискорює швидкість HTML-блоків.
- Відмінні вбудовані інструменти. Для JavaScript фреймворк, Vue.js має чудовий набір інструментів з його новим Vue CLI. З такими речами, як маршрутизація, лінтинг, державний магазин, препроцесори CSS, модульне тестування, PWA, Typescript, ви можете почати новий проект, а в майбутньому ви можете зберегти свої налаштування для використання в інших проектах. Управління своїми проектами, також дозволяється, надаючи користувальницький інтерфейс для додавання функцій вниз по дорозі. До наявного веб-проекта легко додати Vue.JS. Дуже швидко ви можете почати кодування і про системи складання або JSX, ES2015 вам не потрібно багато знати. Фреймворк може бути легким для тих, хто знайомий з JavaScript і HTML.

Якщо ви віддаєте перевагу простоті та гнучкості у своїх фронтальних рамках, то Vue.js — це хороший вибір. Крім того, це найлегший варіант з усіх.

2.3. Платформа Node.js

У двох словах Node.js[7] — це середовище виконання. Що саме мається на увазі Node.js — це JavaScript з відкритим кодом (звідси JS) для створення мережевих додатків. Це дозволяє розробникам Node.js виконувати код на

стороні сервера. Тому Node.js - це легкий, масштабований та швидкий спосіб писати сценарії.

Тепер це добре усталена частина так званої парадигми JavaScript. Це дозволяє та уніфікує розробку додатків, усуваючи потребу в різних мовах. Node.js в основному використовується для створення веб-додатків у режимі реального часу. Однак розробка мобільних додатків також можлива завдяки всій екосистемі Node.js. І його менеджер пакунків - зокрема, NPM.

З Node.js ви можете використовувати код або сценарії, написані також іншими мовами. То як щодо прикладів додатків Node.js? Нижче наведено кілька найбільш відомих проектів та компаній, що використовують Node.js.

PayPal і Node.js. Ви знаходите PayPal у кожному списку програм, створених на Node.js. І це правильно, адже послуга має справлятися з понад 200 мільйонами активних облікових записів користувачів по всьому світі. Початковою проблемою PayPal були розповсюджені команди, які виконували завдання окремо для програм браузера та сервера. Після того, як PayPal прийняв Node.js, розробники використовують єдину мову - JavaScript.

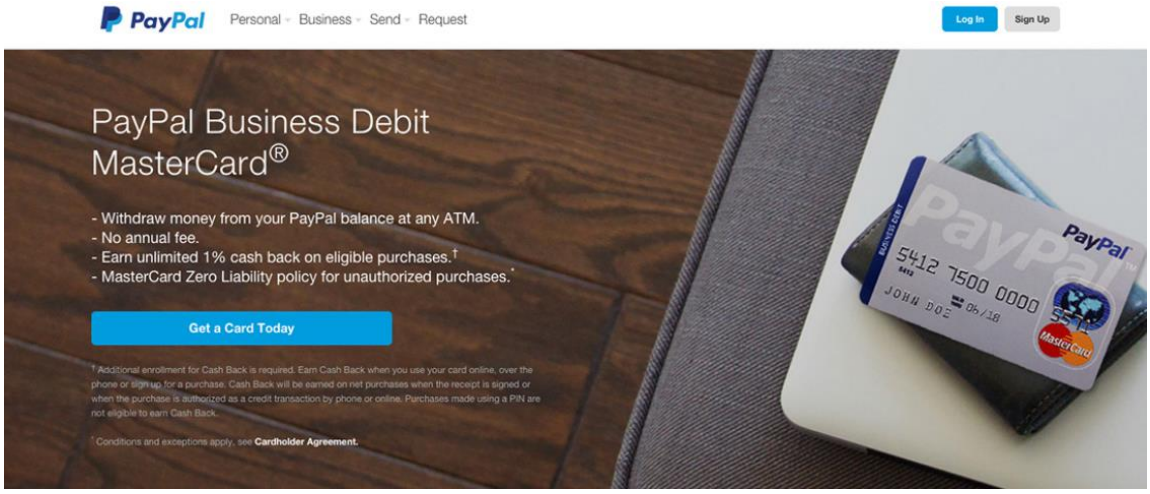


Рис. 2.3.1. PayPal і Node.js.

Згідно із заявою компанії, їх додаток на Node.js було написано в 2 рази швидше, ніж зазвичай. І він містив на 33% менше коду. Мільйони людей, які довіряють їм свої платежі, — це справжній показник сили цього та інших прикладів додатків Node.js.

Ebay і Node.js. Для величезного трафіку Ebay повинен був прийти з перевіреною технологією. Node.js, як частина стека технологій JavaScript, добре підходив для цього гіганта електронної комерції. Після жорстких внутрішніх дискусій інженери Ebay обрали Node. Вирішальним фактором була необхідність зробити якомога більше заявок на Ebay в режимі реального часу.

Рис. 2.3.2. Ebay і Node.js.

Маючи близько 170 мільйонів активних користувачів, додаток Ebay на прикладі Node.js демонструє можливість підтримувати живі з'єднання з серверами. Технічна краса та принцип розгортання на Ebay полягають у наступному. Створіть один раз, розгорніть всюди і автоматизуйте решту. Ebay почав з одного проєкту, і тепер вони переходять до повнофункціонального стека у випадку Node і стають кращим прикладом програми Node.js.

Netflix та Node.js. Netflix — найбільший світовий онлайн відео сервіс, тому вибір ним Node.js говорить про багато. Це також один із найцікавіших прикладів додатків на Node.js. Тому що метою постачальника відео було спеціально оживити користувальницькі інтерфейси. За допомогою проєкту Node.js вони скоротили час збірки та включили налаштування користувачів.

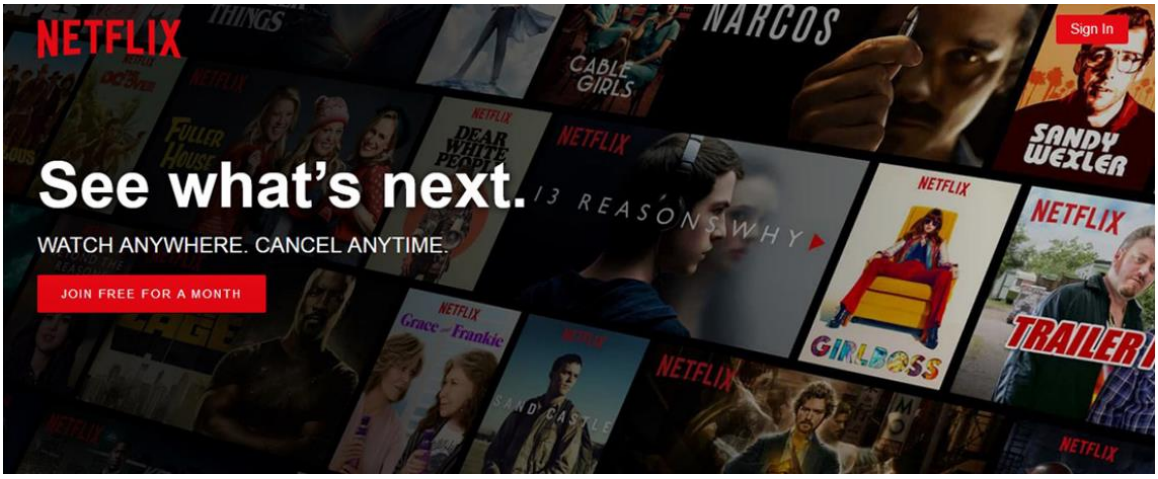


Рис. 2.3.3. Netflix і Node.js.

Крім того, Netflix виграв битву за продуктивність. Компанія повідомляє, що покращила час завантаження програми на 70%. Виконання Node.js виявилося настільки ефективним в Netflix, що вони навіть переміщують до нього шари доступу до даних. Вони також прагнуть писати сценарії виключно як програми Node. Для моніторингу ефекту розробники Netflix використовують метрику TTI - час на інтерактив. Це час між запуском програми та взаємодією з користувачем.

Коротке резюме Node.js:

- Node.js є серверної структурою і є безкоштовним
- Він працює на Windows, Linux, Mac OS і так далі
- Node використовує JavaScript на сервері

Як працює Node.js? Беручи просту задачу відкриття файлу на сервері, послідовність була б:

- Відбувається перехід у файлову систему
- Система готова до запитів
- При відкритті і читанні файлу система відправляє вміст клієнту

Іншими словами, з Node вам не доведеться чекати і ви можете продовжувати виконання наступних завдань. Це одна з причин того, що він такий ефективний. Тепер, що таке Node.js файл:

- Він містить завдання та виконує їх у встановлених подіях
- Подія - це коли хтось намагається отримати доступ до сервера

- Файл слід ініціювати на сервері
- Файли мають розширення ".js"

І останнє, але не менш важливе, що можна зробити з Node.js?:

- Створювати динамічний контент
- Створювати, відкривати та читати файли на сервері
- Збирати та змінювати дані в базі даних

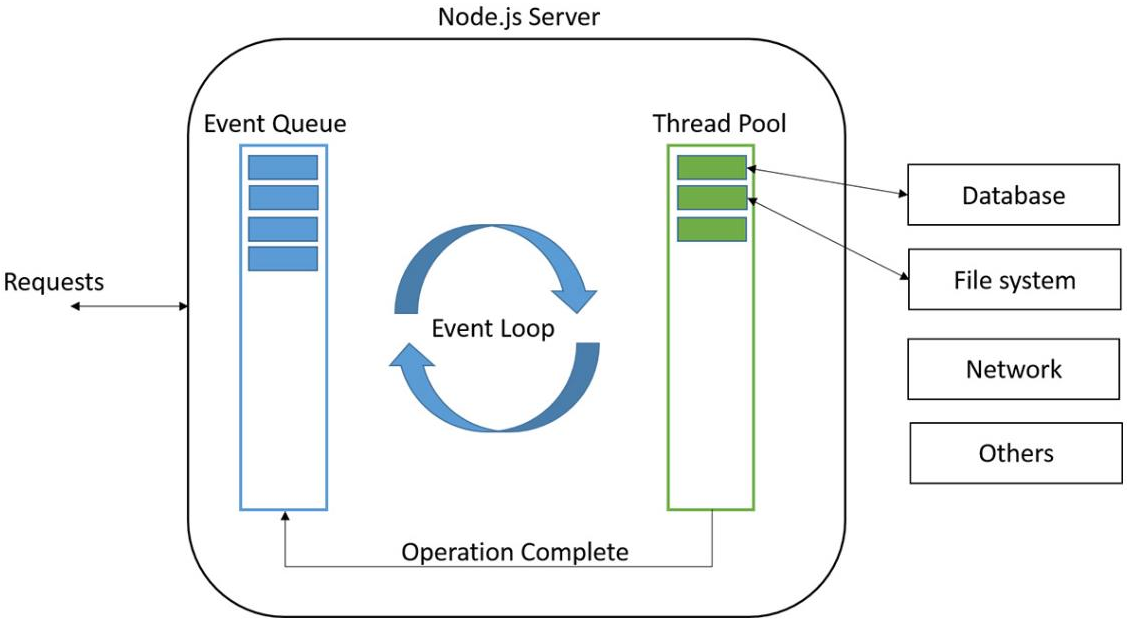


Рис. 2.3.4. Node.js сервер.

Основні причини використання Node.js:

- Добре підходить для початківців розробників, JavaScript простий в освоєнні, багатий фреймворк (Angular, Node, Backbone, Ember)
- Швидкість, завдяки інноваційним технологіям Google і циклу подій
- Можливість зберігати дані у власному форматі JSON (Object notation) у своїй базі даних
- Кілька модулів (NPM, Grunt та ін.) і підтримує співтовариство
- Добре підходить для створення додатків в реальному часі, таких як чати та ігри
- Одиночна вільна кодова база
- Добре підходить для потокової передачі даних, наприклад, для

аудіо та відеофайлів

- Спонсор: Linux Foundation, а також PayPal, Joylent, Microsoft, Walmart
- Широкий вибір варіантів розміщення

2.4. Application Programming Interface (API)

В основних термінах API-інтерфейси просто дозволяють додаткам взаємодіяти один з одним.

Коли люди говорять про "API", вони іноді узагальнюють, а фактично це означає "загальнодоступний веб-API, який повертає дані, ймовірно, в JSON, XML".

API[8] — це навіть не база даних або сервер, це код, який управляє точкою доступу для сервера.

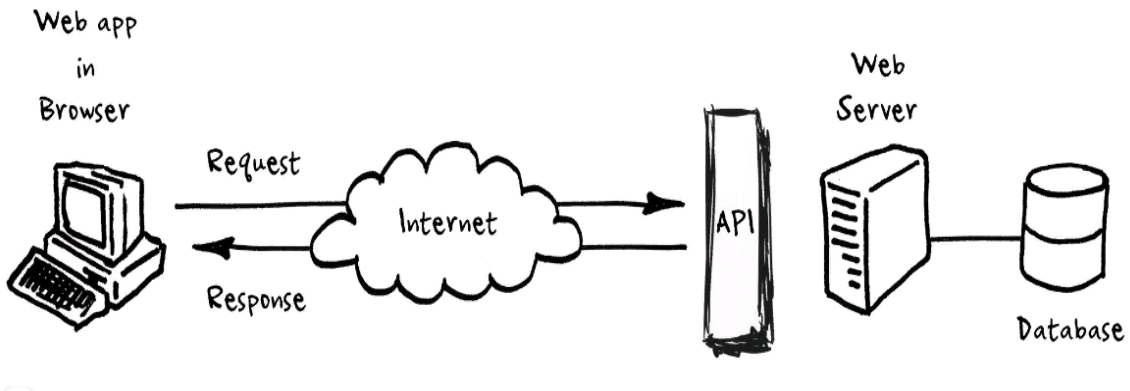


Рис. 2.4.1. Доступ до бази даних через API.

Занадто часто API-інтерфейси розроблялися навколо віддаленого виклику процедур (RPC), так що API-інтерфейси виглядали і відчували себе як локально виконуваний код. Хоча це зробило API-інтерфейси схожими на функції, це також ускладнило їх перехід в мережу.

Попутно багато технологічні стеки намагалися вирішити цю проблему, надаючи розробникам RPC-подібний стек, який намагався приховати деталі. Цей підхід допомагав швидше переміщати програми в інтернет, але залишав розробників писати неефективний код. В середині цього, з'явився впливовий документ, який говорив про створення веб-інтерфейсів API в кращому вигляді: REpresentational State Transfer (REST).

2.4.1 Як REST може допомогти?

REST[9] — це архітектурний стиль, який використовує прості HTTP-виклики для міжмашинної комунікації замість складніших варіантів, таких як CORBA, COM +, RPC або навіть SOAP. Використання REST означає, що ваші виклики будуть засновані на повідомленнях і залежать від стандарту HTTP для опису цих повідомлень.

Використання протоколу HTTP означає, що REST — це простий механізм запиту-відповіді. Кожен запит повертає наступну відповідь. Нижче наведено, як виглядає спрощений запит і відповідь:

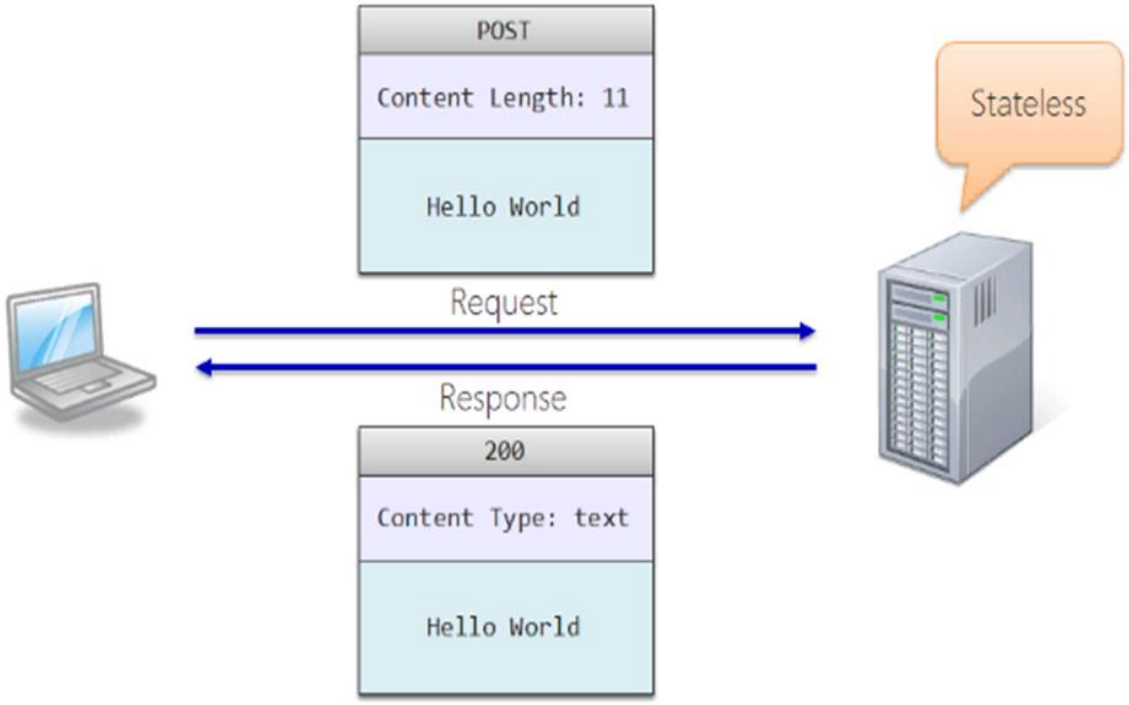


Рис. 2.4.2. Доступ до бази даних через API.

Запити складаються з дієслова (POST, в цьому прикладі), заголовків, що описують повідомлення, і тіла (Hello World, в цьому прикладі). Запит являє собою повідомлення, що описує, що ви хочете, щоб сервер виконав. Аналогічно, відповідь складається з трьох частин: коду стану (200), заголовків, що описують відповідь, і самого тіла.

Команди HTTP описують тип операції:

- GET: витяг ресурсу

- POST: створення ресурсу
- PUT: оновлення ресурсу
- DELETE: видалення ресурсу

В Інтернеті найбільш поширеним дієсловом є GET. Це відбувається тому, що основна мета функції веб-сторінки полягає в запиті різних ресурсів, складових сторінки. В API на основі REST ми використовуємо ці дієслова для опису типів операцій, які ми хочемо.

Це працює, тому що ви не прив'язуєте свій API до своєї технології на стороні клієнта. Можна уявити, що цей API доступний із веб-проєкту на стороні клієнта. Це дозволяє вам створити інфраструктуру для організації з меншими занепокоєннями щодо довгострокового з’єднання з певним стеком на стороні клієнта. Зазвичай сервер живе довше, ніж клієнт.

Інша ключова ідея цієї архітектурної філософії полягає в тому, що сервер підтримує кешування. Ці ідеї дуже важливі для того, як працює REST. Кешування має важливе значення, так як якщо запитується кілька запитів для одного і того ж ресурсу, кешування результату запиту означає, що масштабованість сервера повинна зростати.

Крім того, становище полягає в тому, щоб виклики API не були прив'язані до конкретного сервера, що збільшує ймовірність побудови масштабованої інфраструктури сервера. Оскільки сервер не має статусу, кожен виклик на нього повинен містити всі дані, необхідні для виконання запиту. Протокол HTTP не пов'язаний (немає гарантії того, що ваш запит буде перенесений на один і той же сервер або скільки часу буде тривати між цими дзвінками на сервер).

Отже головними перевагами REST є[10]:

- REST архітектурний зразок в основному легкий за своєю природою.
- Більшість сайтів соціальних мереж, таких як Facebook, використовує послуги REST.

- При обмеженні пропускної здатності веб-сервіс REST має перевагу.
- Легкий і швидкий розвиток.
- Топ сайтів, як Twitter, Yahoo використовує цю схему.
- Розвиток мобільних додатків швидко зростає і для їх взаємодії з сервером використовують схему REST, оскільки це швидше при обробці даних запитів-відповідей.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даном розділі дипломного проєкту я провів дослідження методів створення SPA та інструменти для його розробки. Розглянув архітектуру і роботу SPA, його сильні і слабкі сторони. Було оглянуто основні фреймворки такі як: Angular, React і Vue. Вивчено характеристики і головні переваги кожного, для кращого розуміння в цілому.

Для серверної частини була досліджена платформа Node.js, яка має великі переваги порівнюючи з іншими. Було наведено кілька найбільш відомих проектів та компаній, що використовують Node.js, що доводить ефективність і популярність даної платформи.

Під час проведення даної роботи мною було вибрано оптимальний варіант розробки мого веб-додатку. Таким чином для створення інтерактивних інтерфейсів вибір впав на React, так як він є простим у використанні, для спілкування між датчиками і сервером було вибрано технологію API, а для серверної частини платформу Node.js.

РОЗДІЛ 3.

3. МОДЕЛЮВАННЯ І РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Вибір мови програмування

JavaScript[11] -це динамічна мова комп'ютерного програмування. Вона є легкою і найбільш часто використовується в складі веб-сторінок, реалізація яких дозволяє клієнтського сценарія взаємодіяти з користувачем і створювати динамічні сторінки. Це інтерпретується в мова програмування з об'єктно-орієнтованими можливостями.

JavaScript працює з HTML і CSS для побудови веб-додатків або веб-сторінок. JavaScript підтримується більшістю сучасних веб-браузерів, таких як Google Chrome, Firefox, Safari, Microsoft Edge, Opera та ін. Більшість мобільних браузерів для Android і iPhone тепер підтримують JavaScript.

Інтерфейси прикладного програмування (API) також підтримуються JavaScript, що дає більше функціональності.

Найбільш ранні втілення JavaScript були розроблені в кінці 1990-х років для веб-браузера Netscape Navigator. У той час веб-сторінки були статичними, пропонуючи мало взаємодії з користувачем, крім натискання посилань і завантаження нових сторінок. Вперше JavaScript включив анімацію, адаптацію вмісту і перевірку форми на сторінці.

Протягом багатьох років JavaScript працював тільки на обмеженій кількості браузерів. У перші дні існування програмісти, які хотіли створювати динамічні веб-сайти, часто були змушені вибирати одне сімейство браузерів замість іншого. Це було далеко не ідеально, оскільки робило Інтернет менш доступним для всіх.

JavaScript не був стандартизований і широко прийнятий до 1999 року. Навіть після стандартизації сумісність браузерів залишалася проблемою впродовж більше десяти років[12].

Веб-браузер завантажує веб-сторінку, аналізує HTML і створює те, що відомо як модель об'єкта документа (DOM) з вмісту. DOM являє собою живе уявлення веб-сторінки для коду JavaScript[13].

Потім браузер захоплює все, що пов'язано з HTML, наприклад: зображення і файли CSS. Інформація CSS надходить від аналізатора CSS.

HTML та CSS об'єднуються DOM для створення веб-сторінки в першу чергу. Потім двигун JavaScript браузерів завантажує файли JavaScript та вбудований код, але не виконує код негайно. Він очікує завершення завантаження HTML і CSS.

Після цього JavaScript виконується в тому порядку, в якому був написаний код. Це призводить до того, що DOM оновлюється JavaScript і візуалізується браузером.

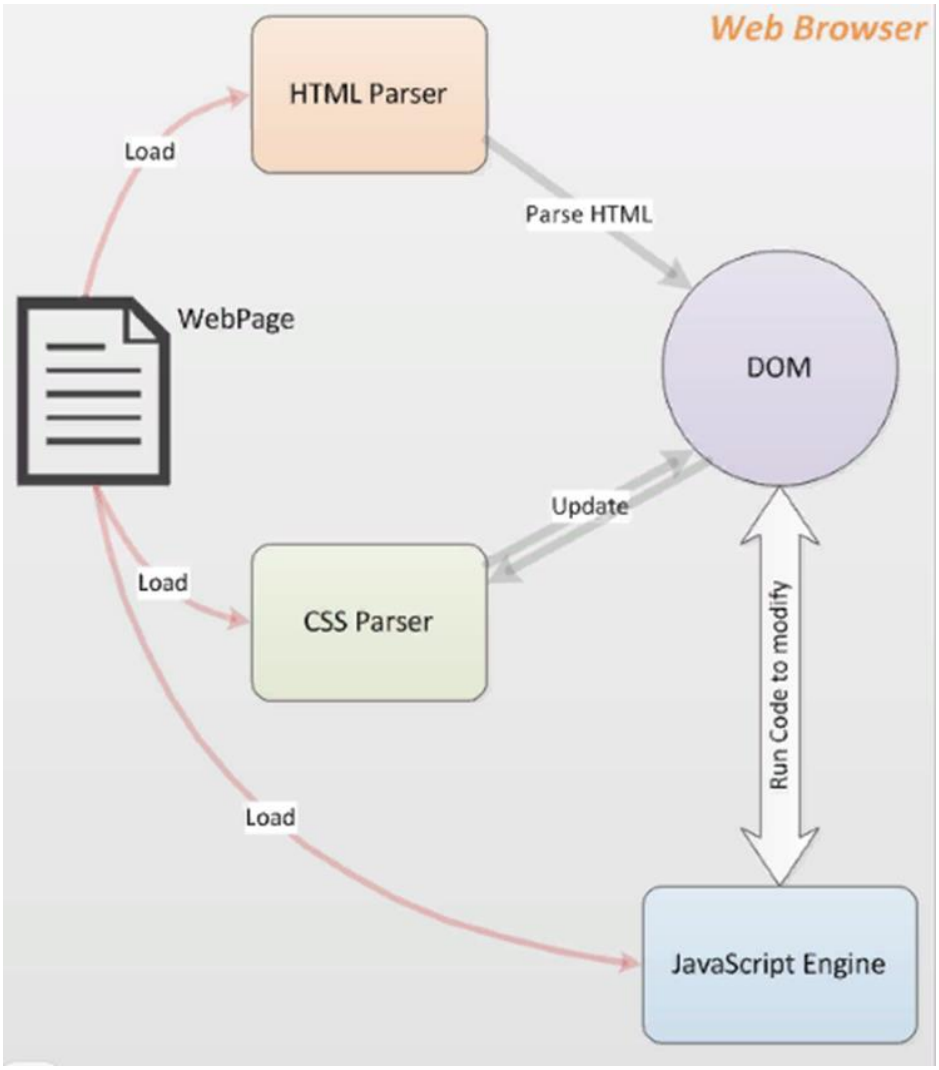


Рис. 3.1. Робота JavaScript.

Порядок тут дуже важливий. Якщо JavaScript не дочекався завершення HTML і CSS, він не зможе змінити елементи DOM.

Що можна робити з JavaScript:

- Оголошення змінних.
- Зберігання та вилучення значень.
- Визначення та виклик функцій, включаючи функції зі стрілками.
- Визначення об'єктів і класів JavaScript.
- Завантаження та використання модулів.
- Запис обробників подій, які відповідають на події click.
- Написання серверного коду.

Переваги використання JavaScript полягають у наступному:

- Менше взаємодії з сервером – перевіряє введення перед відправкою сторінки на сервер. Це економить трафік сервера, що означає менше навантаження на сервер.
- Негайна зворотній зв'язок з відвідувачами − не потрібно чекати перезавантаження сторінки, щоб побачити, якщо забули ввести щось.
- Підвищена інтерактивність – створює інтерфейси, які реагують, коли користувач наводить на них мишкою або активує їх за допомогою клавіатури.
- Більш багаті інтерфейси – використання JavaScript для включення таких елементів, як компоненти перетягування і повзунки, щоб дати багатий інтерфейс для відвідувачів сайту.

Обмеження JavaScript:

- Клієнтський JavaScript не дозволяє читати або записувати файли. Це було збережено з міркувань безпеки.
- JavaScript не можна використовувати для мережевих додатків, оскільки така підтримка відсутня.

- JavaScript не має ніяких багатопоточних або багатопроцесорних можливостей.

3.2 Вибір бази даних

База даних (БД), в найзагальнішому розумінні, - це організований збір даних. Більш конкретно, база даних - це електронна система, яка дозволяє легко отримувати доступ, маніпулювати та оновлювати дані.

Іншими словами, база даних використовується організацією як метод зберігання, управління та отримання інформації. Сучасними базами даних керують за допомогою системи управління базами даних (СУБД).

3.2.1. Системи управління базами даних (СУБД)

3.2.1.1. MySQL

MySQL[14] - одна з найпопулярніших систем управління реляційними базами даних, створена в 1995 році і зараз керується Oracle. Ця система баз даних з відкритим кодом має величезну базу користувачів та чудову підтримку, і вона добре працює з більшістю бібліотек та фреймворків. Це безкоштовно, але пропонує додаткові функціональні можливості за фіксовану ціну.

Рис. 3.2. Логотип MySQL.

Розробники можуть встановлювати та використовувати MySQL, не витрачаючи довгих годин на його налаштування. Більшість завдань можна виконати в командному рядку. Це добре структурована база даних з регулярними оновленнями.

MySQL відмінно працює зі структурованими даними на базовому рівні. Але якщо ви плануєте розглянути можливість масштабування вашого продукту в майбутньому, вам може знадобитися додаткова підтримка, яка коштує досить дорого. Також потрібно багато часу для створення покрокових резервних копій або зміни архітектури даних в MySQL, тоді як його конкуренти можуть це робити автоматично.

Uber, Facebook, Tesla, YouTube, Netflix, Spotify, Airbnb та багато інших компаній використовують MySQL для своїх послуг.

3.2.1.2. PostgreSQL

PostgreSQL — об’єктно-реляційна база даних, яка схожа на реляційні бази даних, але всі дані представлені у вигляді об’єктів замість стовпців та рядків.

Рис. 3.3. Логотип PostgreSQL.

PostgreSQL є ідеальним рішенням для великих систем, оскільки він масштабований і призначений для обробки терабайтів даних. Ієрархія ролей для підтримки дозволів користувача означає розширену безпеку.

На відміну від MySQL, PostgreSQL є абсолютно безкоштовним у використанні. Його відкритий характер означає, що всю документацію та підтримку надають захоплені волонтери.

3.2.1.3. MongoDB

MongoDB — це база даних. У MongoDB всі дані зберігаються в документах BSON (Binary JSON). Завдяки цьому, дані можуть легко

передаватися між веб-додатками та серверами у читаному для людини форматі.

MongoDB має бортову реплікацію, що забезпечує високу масштабованість та доступність. Автоматичне сегментування означає, що ви можете легко поширювати дані на сервери, підключені до вашої програми. Взагалі, MongoDB — це чудове рішення для боротьби з масовими неструктурованими наборами даних. Він може лежати в основі більшості великих систем даних не тільки як оперативний сховище даних в режимі реального часу, але і в автономному режимі.

Рис. 3.4. Логотип MongoDB.

Але є кілька підводних каменів цієї платформи баз даних. Він зберігає ключові імена для кожної пари цінностей, збільшуючи використання пам'яті. Також немає жодних зовнішніх ключових обмежень для забезпечення узгодженості, і ви можете виконувати вкладення не більше 100 рівнів.

3.2.1.4. OracleDB

OracleDB, розроблений у 1977 році, залишається відомим і надійним рішенням. Він займає перше місце в рейтингу DB-Engines.

Причини популярності OracleDB:

- Розробники зазначають, що OracleDB рідко знижується і отримує регулярні оновлення.

- Він добре масштабує і може обробляти надзвичайно великий обсяг даних. В даний час Oracle виводить всі свої продукти та послуги в хмару, що призводить до більшої гнучкості.
- Це безпечно, ретельно дотримуючись сучасних стандартів безпеки (включаючи відповідність PCI) та пропонує добре шифрування конфіденційних даних.
- Він керує пам'яттю дуже ефективно і легко обробляє складні операції. Крім того, він ефективно керує та організовує різноманітні сторонні інструменти.
- Він перевершує інші рішення щодо швидкості доступу до даних через мережу.

Рис. 3.5. Логотип OracleDB.

Але у OracleDB є і мінуси:

- Завдяки популярній тепер СУБД, OracleDB також є однією з найдорожчих. Ліцензія на процесор для стандартної версії обійдеться вам у $ 17 500 за одиницю.
- Oracle має надто складну документацію і не має хороших посібників. Незважаючи на те, що підтримка клієнтів дуже корисна, деякі розробники скаржаться на тривалий час реакції.
- Ці фактори роблять OracleDB ідеальним рішенням для великих підприємств, яким потрібно зберігати великий обсяг даних.

Малому та середньому бізнесу слід шукати більш рентабельні альтернативи.

3.2.1.5. MariaDB

MariaDB[15] має цікаву історію за своїм створенням. Зокрема, MariaDB є розгалуженням MySQL і підтримується деякими початковими розробниками MySQL. Виникали занепокоєння з приводу придбання MySQL Oracle (корпоративного), і тому MariaDB народилася в 2009 році. З того часу проект продовжував розвиватися і розвиватися, його сьогодні використовують приблизно 17% професійних розробників.

Рис. 3.5. Логотип MariaDB.

Одне з чудових речей про MariaDB - це роздрібність оригінального MySQL, це те, що ви можете легко і безпечно переходити від однієї системи баз даних до іншої. Більшість синтаксисів командного рядка залишаються в основному однаковими.

Тож вам може бути цікаво, у чому сенс вибору MariaDB над більш популярною системою MySQL? Давайте розглянемо деякі основні моменти, які демонструють, як MariaDB може покращити ваш досвід управління даними, коли мова йде про бази даних.

Двигуни для зберігання. MariaDB поставляється з десятками двигунів зберігання даних (Cassandra, TokuDB тощо), які дозволяють більш ефективно зберігати дані різних типів. Це також означає, що доступ до зазначених типів даних стає менш залежним від продуктивності сервера.

Швидша індексація та кешування. Унікальний механізм зберігання даних, який називається "Пам'ять", на 25% швидше для команд INSERT, ніж для MySQL. Це набуває значущості, оскільки кількість інформації, яку ви зберігаєте у вашій базі даних, збільшується.

Система плагінів. MariaDB підтримує використання плагінів, програмних компонентів, які можуть бути додані до основного програмного забезпечення без необхідності перебудови сервера MariaDB з вихідного коду. Тому плагіни можна завантажувати при запуску, або завантажувати та вивантажувати, поки сервер працює без перешкод.

Основний двигун сумісний з усіма основними операційними системами, включаючи Windows, Linux і навіть OSX.

Протягом останніх кількох років такі компанії, як Wikipedia, Google та численні банки світового класу, прийняли MariaDB як основний вибір системи баз даних. Це свідчить не тільки про надійність MariaDB, але і про довіру великим брендам до проекту.

3.3. Середовище розробки WebStorm

WebStorm - це інтегроване середовище розробки (IDE), розроблене спеціально для сучасного JavaScript. Ця технологія є легкою, але всеосяжною, оскільки пропонує рішення для складної розробки на стороні клієнта та розробки на сервері з Node.js.

Створений розробником програмного забезпечення для продуктивності JetBrains, WebStorm використовує потужну екосистему JavaScript. Розробники можуть використовувати інтелектуальне завершення коду, виявлення помилок у реальному часі, рефакторинг JavaScript та навігацію, таблицю стилів мови, TypeScript та провідні рамки.

Користувачі можуть легко налагоджувати програми Node.js на стороні клієнта. Технологія розміщує правильні точки вихідного коду, встановлює годинник, досліджує змінні та дзвінки та використовує інтерактивну консоль.

Безшовна інтеграція інструментів дозволяє розробникам максимізувати вбудовані вкладиші, тестові бігуни, клієнт REST та інші глибоко інтегровані властивості. Функціональні можливості тестування блоків дозволяють користувачам налагоджувати тести запускати Mocha, Jest, Karma та Protractor. Стани тестування миттєво доступні як у перегляді дерев, так і в редакторі.

Завдяки інтеграції WebStorm з VCS розробники можуть скористатися простим уніфікованим інтерфейсом для роботи з Git, GitHub, Mercurial тощо. Користувачі можуть фіксувати файли, переглядати зміни та вирішувати конфлікти за допомогою візуального інструменту, інтегрованого в IDE.

На рисунку 3.6 зображено середовище розробки WebStorm з відкритим вікном SensorTable.

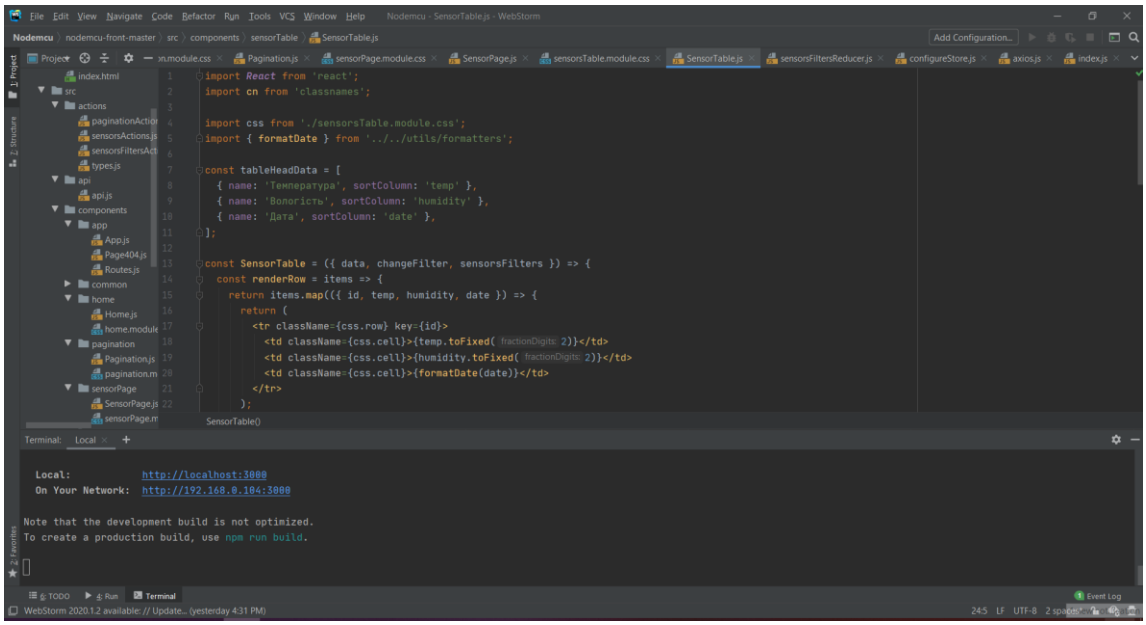


Рис. 3.6. Вигляд середовище розробки WebStorm.

WebStorm надає інтуїтивно зрозумілу допомогу в кодуванні JavaScript і мов компільованого JavaScript, Node.js, HTML та CSS. Розробники можуть насолоджуватися доповненням коду, широкими навігаційними функціями, розпізнаванням помилок на ходу та переробкою для всіх цих мов. Додаток пропонує розширену допомогу в кодуванні для Angular, React, Vue.js та Meteor. WebStorm також підтримує React Native, PhoneGap, Cordova та Ionic для мобільних розробок і розробляє для сервера з Node.js.

Сотні інтегрованих перевірок позначають усі можливі проблеми безпосередньо як тип користувача, а потім пропонує швидко виправити варіанти. Це забезпечує якість розробленого проекту.

WebStorm має вбудовані функції для налагодження, тестування та відстеження ваших клієнтських та Node.js додатків. Завдання стають набагато простішими, оскільки для них потрібна лише мінімальна конфігурація. Користувачі можуть вказувати точки перерви, переходити через код і оцінювати вирази - все, не виходячи з IDE. WebStorm також пропонує шпигун-js, вбудований інструмент, який допомагає відстежувати коди JavaScript. Файли пов'язані з викликами функцій, і потенційні проблеми неполадок визначаються для підвищення ефективності.

WebStorm легко інтегрується з інструментами верхнього командного рядка для веб-розробки. Це забезпечує спрощений досвід розробки, що збільшує продуктивність, не використовуючи командний рядок. Інструменти збирання відображаються у простому уніфікованому інтерфейсі для виконання завдань Grunt, Gulp або npm.

Розробники можуть працювати над іншими програмами за допомогою вбудованих шаблонів WebStorm. Крім популярних, таких як Експрес або Веб-стартерний набір, користувачі можуть отримати доступ до більшої кількості генераторів проектів за допомогою інтеграції WebStorm з Yeoman.

WebStorm дуже настроюється, оскільки його можна налаштувати відповідно до будь-якого стилю кодування розробника. Користувачі можуть налаштувати ярлики, шрифти та візуальні теми до вікна інструментів та макета редактора.

3.4. Розробка програмного продукту

Для створення веб-додатку для відображення температури і вологи в приміщенні було вибрано мову програмування JavaScript, платформу Node.js для серверної частини, та фреймворк React для візуальної частини, так як вони є сумісними і показують свою ефективність для вирішення схожих задач.

Середовище розробки було вибрано WebStorm, так як він виявився найзручнішим і має всі необхідні інструменти для розробки даної системи.

Для повноцінного функціонування даної системи необхідно розробити наступні функції на серверній стороні. Вони виконують наступний необхідний функціонал:

- Отримати дані усіх датчиків
- Отримати дані одного датчика
- Записати дані

Розглянемо більш детально реалізацію цих функцій. Для спілкування з датчиками було вибрано створення API. Щоб отримати інформацію з усіх датчиків, за допомогою запита get на посилання api/sensor. Дана функція підтримує параметри:

- SortDir – напрямок сортування
- SortColumn – задає назву колонки для сортування

```
// ========== GET ============

router.get('/test', (req, res) => {
  res.status(200).send({
    message: 'api test sensor',
    status: 'ok',
  });
});

/**
 * @param {number | string} page - page
 * @param {string} sortDir - sort direction
 * @param {string} sortColumn - sorted column name
 */
router.get('/sensor', (req, res) => {
  dbPool.getConnection() Promise<PoolConnection>
    .then(connection => {

      const page = req.query.page || 1;

      const sortDirection = req.query.sortDir || 'DESC';
      const escapedSortDirection = sortDirection.toLowerCase() === 'desc' ? 'DESC' : 'ASC';
      const sortColumn = req.query.sortColumn || 'date';


      const skippedItems = (page - 1) * LIMIT_PER_PAGE;

      const selectQuery = `
        SELECT * FROM sensors
        ORDER BY ${connection.escapeId( identifier: `sensors.${sortColumn}`)} ${escapedSortDirection}
```

Рис. 3.7. Функція отримання всіх даних з датчиків.

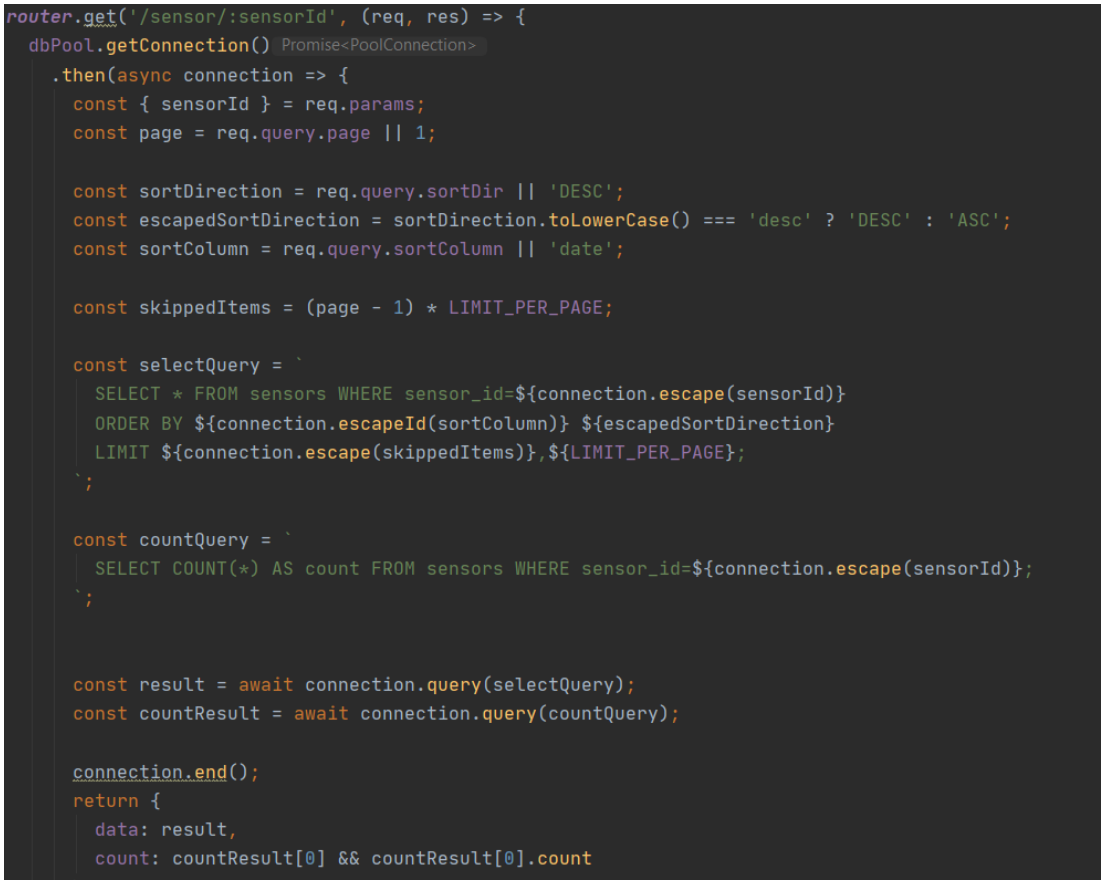
На рисунку 3.8 зображено функцію запису в базу даних інформації отриманої з датчиків. Функція отримує наступні параметри:

- Temp – параметр температури
- Humidity – параметр вологи
- Sensor_id – параметр номера кімнати

```
// =========== POST ===============

router.post('/sensor/:sensorId', (req, res) => {
  dbPool.getConnection() Promise<PoolConnection>
    .then(connection => {
      const { sensorId } = req.params;
      const { temp, humidity } = req.body;

      const selectQuery = `
        INSERT INTO sensors (temp, humidity, date, sensor_id)
        VALUES(${connection.escape(temp)}, ${connection.escape(humidity)},
        ${connection.escape(new Date())}, ${connection.escape(sensorId)});
      `;

      const result = connection.query(selectQuery);
      connection.end();
      return result;
    }) Promise<any>
    .then(result => {
      const data = result[0];
      res.status(200).send({
        data
      });
    }) Promise<any>
    .catch(err => {
      console.log(err);
      res.status(500).send({ error: err.message });
    });
});
```

Рис. 3.8. Функція запису в базу даних.

За допомогою ще одного get запита з відомим параметром sensorId, ми можемо нарешті отримати точні дані з вибраного датчика. Реалізація відповідного коду наведена нижче на рисунку 3.9.



```
router.get('/sensor/:sensorId', (req, res) => {
  dbPool.getConnection() Promise<PoolConnection>
    .then(async connection => {
      const { sensorId } = req.params;
      const page = req.query.page || 1;

      const sortDirection = req.query.sortDir || 'DESC';
      const escapedSortDirection = sortDirection.toLowerCase() === 'desc' ? 'DESC' : 'ASC';
      const sortColumn = req.query.sortColumn || 'date';

      const skippedItems = (page - 1) * LIMIT_PER_PAGE;

      const selectQuery = `
        SELECT * FROM sensors WHERE sensor_id=${connection.escape(sensorId)}
        ORDER BY ${connection.escapeId(sortColumn)} ${escapedSortDirection}
        LIMIT ${connection.escape(skippedItems)},${LIMIT_PER_PAGE};
      `;

      const countQuery = `
        SELECT COUNT(*) AS count FROM sensors WHERE sensor_id=${connection.escape(sensorId)};
      `;


      const result = await connection.query(selectQuery);
      const countResult = await connection.query(countQuery);

      connection.end();
      return {
        data: result,
        count: countResult[0] && countResult[0].count
```

Рис. 3.9. Функція отримання даних з одного датчика.

Зверніть увагу для того щоб забезпечити безпечне функціонування бази даних усі параметри екрануються. Це необхідно для захисту бази даних від SQL ін'єкцій.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було розглянуто мову програмування JavaScript, як мову для написання проєкту. Переглянуто доступні бази даних, проведено аналіз і вирішено вибрати MariaDB для зберігання даних з датчиків.

Середовище для розробки було вибрано WebStorm, так як в ньому є всі необхідні інструменти для розробки проєкта. Було розроблено програмну частину для збору даних з датчиків і запису їх в базу даних. Також було зроблено фронтову частину проєкту, яка буде представлена в наступному розділі.

РОЗДІЛ 4.

4. ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ

4.1. Інструкція користувача

Після того як система готова до роботи, сервер налаштований, пристрої зчитування температури і вологи підключені до розетки і є доступ до Wi-fi. Оскільки дані з датчиків приходять постійно, сервер має працювати весь час, для того щоб покази могли бути записаними в базу даних, з яких потім потраплять в таблицю на веб-сторінці, відповідно до кімнати в якій проводилось вимірювання.

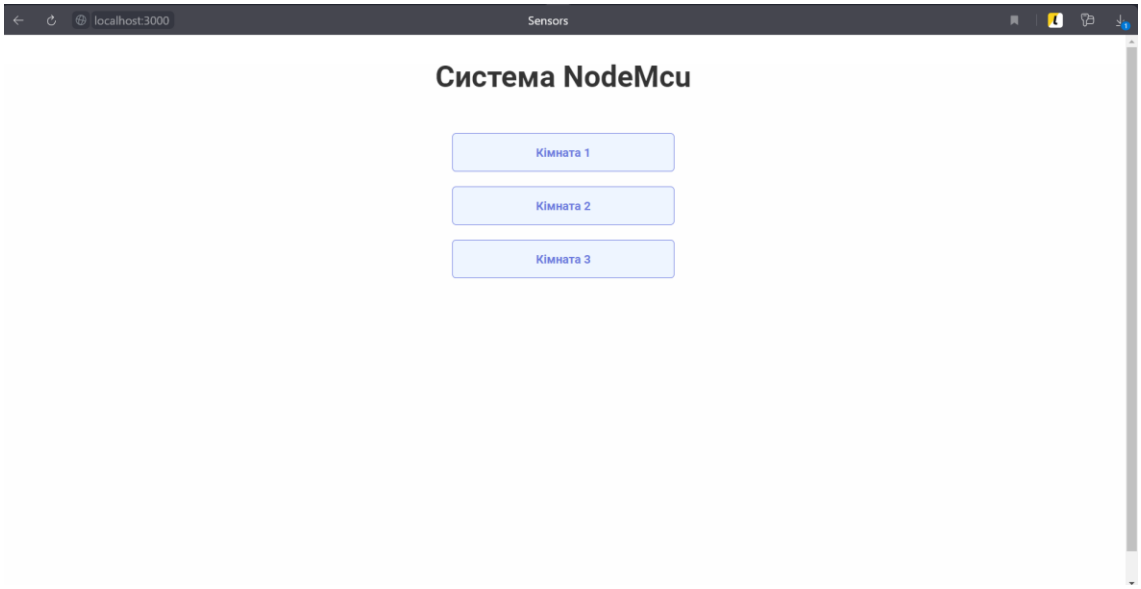


Рис. 4.1. Вигляд головної сторінки системи.

На рисунку 4.1 зображено вигляд головного вікна веб-додатку. На цю сторінку можна додати необхідну кількість кімнат в яких потрібно проводи виміри, а також змінювати назву в залежності від бажання.

Все що потрібно для того щоб отримати доступ до сайту — це підключення до локальної мережі, до якої підключені датчики вимірювання і пристрій в якому є доступ до веб-браузера.

Далі користувач може вибрати кімнату, за допомогою кліку на відповідну кнопку. Відкривається нове вікно яке зображено на рисунку 4.2.

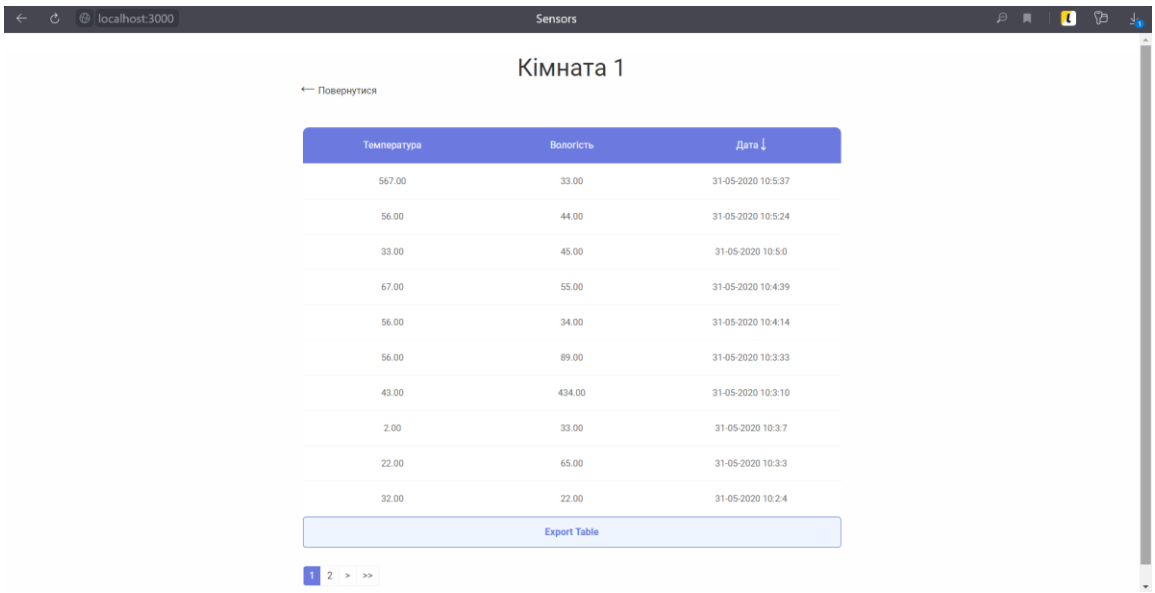


Рис. 4.2. Вигляд сторінки з результатами кімнати 1 системи.

На рисунку 4.2 можна побачити велику таблицю в яку записуються всі дані які приходять з датчиків. Основні поля: температура, вологість, дата.

За кожним з полів можна робити фільтрацію за допомогою функціонального елемента "Стрілка". Для фільтрації потрібно натиснути на назву того поля за яким хочеться зробити фільтрацію. На рисунку 4.3 виділено цей функціональний елемент.

Рис. 4.3. Вигляд функціонального елемента "Стрілка".

Також на сторінці присутній функціональний елемент "Повернутися", який знаходиться зверху в лівому кутку, за допомогою якого можна повернутися на головну сторінку, для вибору іншої кімнати наприклад. На рисунку 4.4 можна побачити елемент більш детально.

Рис. 4.4. Вигляд функціонального елемента "Повернутися".

Не менш важливим є можливість експорту показів на свій девайс в конкретну папку в ексель форматі. Далі з показами можна робити все що заманеться, наприклад відправити електронною поштою або за допомогою месенджерів, надрукувати, форматувати або просто вести статистику.

Кнопка "Export table" знаходиться внизу під таблицею, показано на рисунку 4.

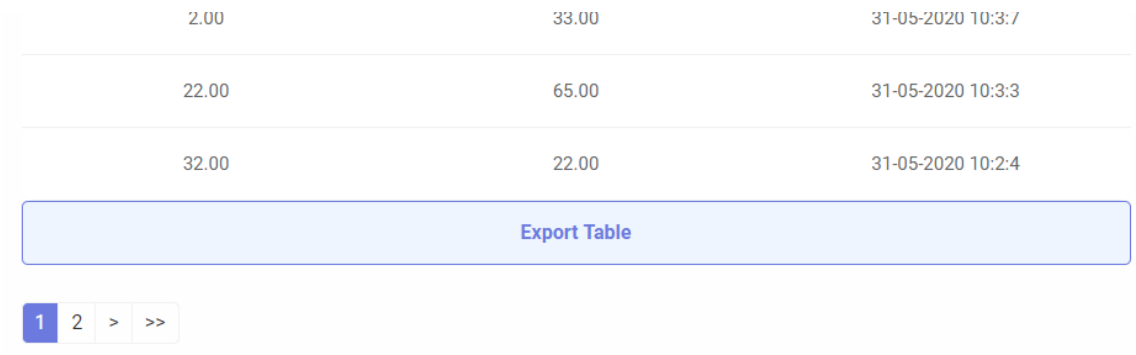


Рис. 4.5. Вигляд кнопки "Export Table".

Також на сторінці присутня пагінація для швидкого переходу між показами однієї кімнати. Більш детально показано на рисунку 4.6.

Рис. 4.6. Вигляд кнопок пагінації.

Система є універсальна, адже отримати доступ до даних можна маючи будь-який девайс з браузером під рукою і доступом до Wi-fi мережі. А також є легкою у використанні, із зрозумілим інтерфейсом та необхідним функціоналом, якого більш як достатньо для систем такого типу.

ВИСНОВКИ ДО РОЗДІЛУ 4

В даному розділі було детально описано роботу системи, з чіткими інструкціями її використання. Система є легкою у використанні і має зрозумілий кожному інтерфейс.

Було наведено достатньо скріншотів вікон програми, з детальним описом кожної кнопки і елементів які знаходяться на сторінці. Система не потребує встановлення додатку, достатньо стандартного браузера і підключення до Wi-fi мережі.

ВИСНОВКИ

Під час виконання даної бакалаврської роботи була розроблена та перевірена робота системи з вимірювання та відображення на веб-сторінці даних з датчиків вимірювання температури і вологи. Було проведено глобальне дослідження існуючих технологій розробки проєкту такого рівня, та вибір найкращих в залежності від потреб.

Розглянуто сильні і слабкі сторони роботи SPA. Було досліджено такі фреймворки як: Angular, React і Vue. В кінцевому результаті перевагу отримав React, який найкраще підходить для таких цілей.

Досліджена платформа Node.js, яка має великі переваги порівнюючи з іншими для серверної частини.

Було розроблено систему яка працює і не потребує доробок, хоча в майбутньому можна було б додати нових функцій таких як:

- Додати графіки температури та вологи за місяць чи рік
- Додати можливість відправлення повідомлення смс або через месенджери про критичну температуру в тій чи іншій кімнаті
- Додати можливість автоматичної відправки статистики на вказану електронну пошту
- Зробити інтерфейс більш привабливим і зрозумілим

В кінці розробки програмного продукту, система була протестована, перевірена на наявність багів, коректне відображення даних по кімнатах. Була запропонована інструкція користувача для швидкого розуміння, як користуватися системою. Система була розроблена таким чином щоб була зрозуміла навіть недосвідченому користувачу.

Система була представлена на кафедрі ОТ і має перспективу на встановлення на ній, для вимірювання температури і вологи в кабінетах та контролю за цими параметрами.

Так як проєктом вже зацікавились, він є перспективним і його можна розвивати в майбутньому для покращення його роботи, якщо встановлених функцій не буде достатньо для роботи.

Після проведення порівняння з аналогами, система поступається по функціональним можливостям, але все одно вона є рентабельною, адже коштує в декілька разів дешевше ніж аналоги на ринку.

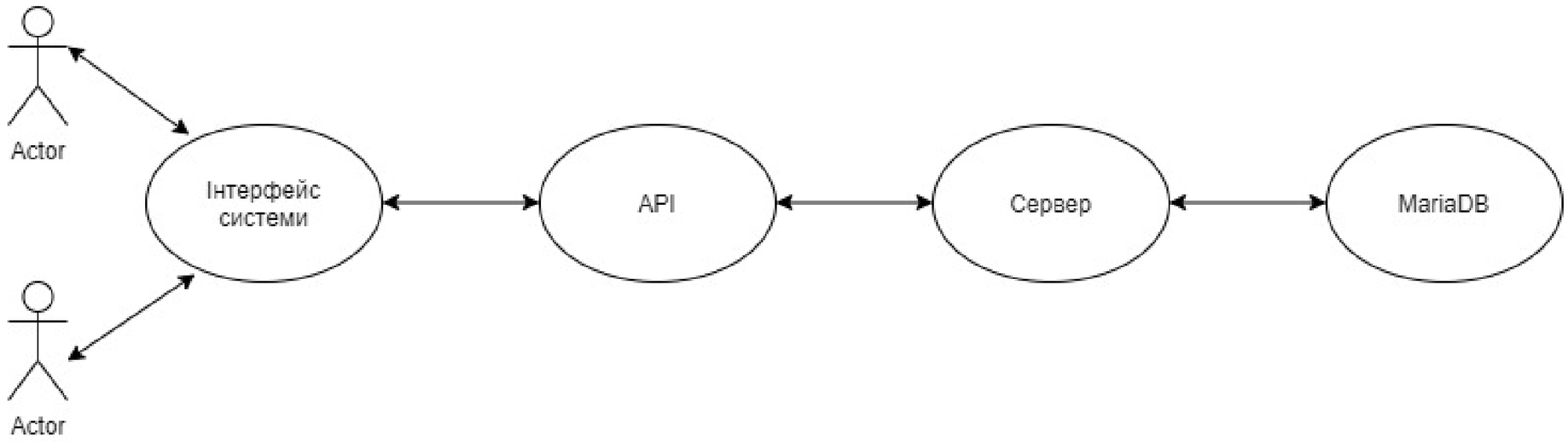
В ході даної роботи були виконані всі поставлені цілі та отриманий гарний результат.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

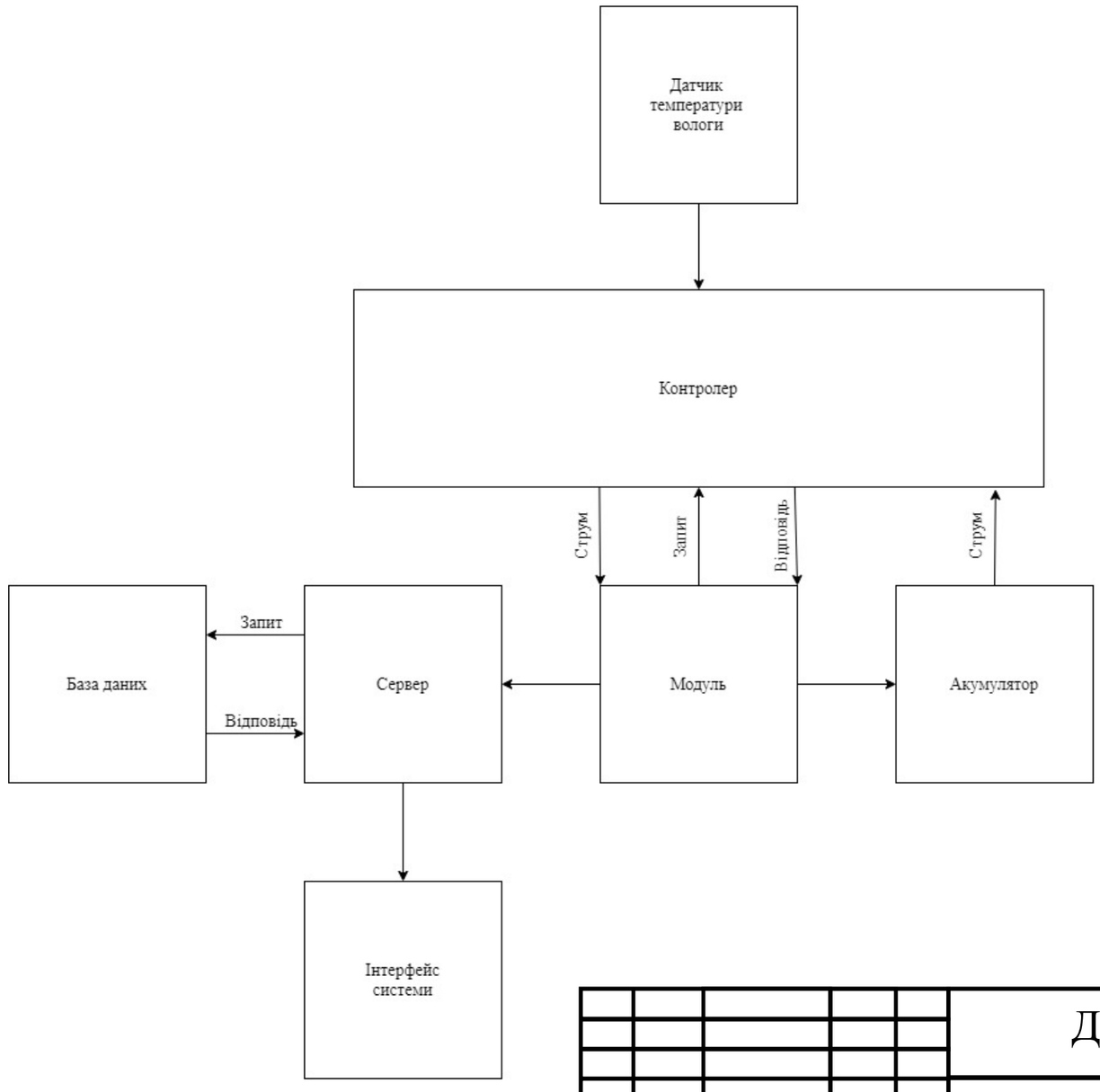
1. Robert Gibb. What is a Web Application? [Електронний ресурс] / Robert Gibb. – 2016. – Режим доступу до ресурсу: https://blog.stackpath.com/web-application/.
2. What is Web Application? [Електронний ресурс] – Режим доступу до ресурсу: https://www.educba.com/what-is-web-application/.
3. Zee Gimon. What is a Single Page Application? [Електронний ресурс] / Zee Gimon – Режим доступу до ресурсу: https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications.
4. DreamToIPO. What makes AngularJS a great framework for Single Page Application? [Електронний ресурс] / DreamToIPO. – 2017. – Режим доступу до ресурсу: https://medium.com/@dreamtoipo/what-makes-angularjs-a-great-framework-for-single-page-application-c7c023c84c08.
5. Tutorial: Intro to React [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/tutorial/tutorial.html.
6. Why and Where Should you Use React for Web Development? [Електронний ресурс] – Режим доступу до ресурсу: https://www.simform.com/why-use-react/.
7. Victor Mangur. 10 best Node.js app examples for enterprises, with metrics [Електронний ресурс] / Victor Mangur – Режим доступу до ресурсу: https://thinkmobiles.com/blog/node-js-app-examples/.
8. Perry Eising. What exactly IS an API? [Електронний ресурс] / Perry Eising. – 2017. – Режим доступу до ресурсу: https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f.
9. Shawn Wildermuth. What is REST API and Why Use It [Електронний ресурс] / Shawn Wildermuth. – 2015. – Режим доступу до ресурсу: https://www.pluralsight.com/blog/tutorials/representational-state-transfer-tips.
10. GearBrain. What is the conclusion for a RESTful web service? [Електронний ресурс] / GearBrain – Режим доступу до ресурсу: https://www.quora.com/What-is-the-conclusion-for-a-RESTful-web-service.

11. What is JavaScript, and why is it important? [Електронний ресурс] – Режим доступу до ресурсу: https://www.bigcommerce.com/ecommerce-answers/what-javascript-and-why-it-important/.
12. JavaScript - Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/javascript/javascript_overview.htm.
13. Anthony Grant. What Is JavaScript and How Does It Work? [Електронний ресурс] / Anthony Grant. – 2019. – Режим доступу до ресурсу: https://www.makeuseof.com/tag/what-is-javascript/.
14. Alex Ivanovs. MongoDB vs MariaDB vs MySQL [Електронний ресурс] / Alex Ivanovs. – 2019. – Режим доступу до ресурсу: https://geekflare.com/mongodb-vs-mariadb-vs-mysql/.
15. Liliia Harkushko. Vital Things to Consider When Choosing a Database for Your App [Електронний ресурс] / Liliia Harkushko – Режим доступу до ресурсу: https://yalantis.com/blog/how-to-choose-a-database/.

ДОДАТОК А



					ДП 6322 03.000 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата	*UML діаграма взаємодії користувача з системою*	*Літера* / *Маса* / *Масштаб*
Розроб.		Орихіський Є.Р				
Перевір.		Сімоненко В.П.			*Схема структурна*	*Аркуш 1* / *Аркушів 1*
Т. контр.						
Н. контр.		Сімоненко В.П.				*НТУУ "КПІ" ФІОТ*
Затв.						*Група ІО-63*



					ДП 6322 04.000 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	*Схема взаємодії компонентів системи*	*Літера* / *Маса* / *Масштаб*
Розроб.		Орихіський Є.Р			*Схема функціональна*	
Перевір.		Сімоненко В.П.				*Аркуш 1* / *Аркушів 1*
Т. контр.						
						НТУУ "КПІ" ФІОТ
Н. контр.		Сімоненко В.П.				*Група ІО-63*
Затв.						

					ДП 6322 05.000 ДЗ	
Зм.	Арк.	№ докум.	Підпис	Дата	*Алгоритм роботи системи* *Схема принципова*	*Літера* / *Маса* / *Масштаб*
Розроб.		Орихіський Є.Р.				
Перевір.		Сімоненко В.П.				Аркуш 1 / Аркушів 1
Т. контр.						
Н. контр.		Сімоненко В.П.				*НТУУ "КПІ" ФІОТ* *Група ІО-63*
Затв.						